Oracle® Database

High Availability Overview and Best Practices

F46646-24
June 2024

Oracle Database High Availability Overview and Best Practices,

F46646-24

Contents

3 Features for Maximizing Availability

15 Configure and Deploy Oracle Data Guard

16 Tune and Troubleshoot Oracle Data Guard

23 On-Premises: Configuring Oracle GoldenGate Hub

24 On-Premises: Oracle GoldenGate Microservices Architecture with Oracle Real Application Clusters Configuration Best Practices

30 Oracle Maximum Availability Architecture in Oracle Exadata Cloud Systems

31 Oracle Maximum Availability Architecture for Oracle Database@Azure

32 Oracle Data Guard Hybrid Cloud Configuration

Preface

This book introduces you to Oracle best practices for deploying a highly available database environment, and provides best practices for configuring the Oracle MAA reference architectures.

Part 1 provides an overview of high availability and helps you to determine your high availability requirements. It describes the Oracle Database products and features that are designed to support high availability and describes the primary database architectures that can help your business achieve high availability.

Part 2 describes the best practices for configuring a highly available Oracle database, using features provided with Oracle Database, which lets you achieve MAA Bronze reference architecture service levels

Part 3 describes the best practices for configuring a highly available Oracle database using Oracle Data Guard for replication and data protection, which lets you achieve MAA Gold reference architecture service levels.

This preface contains these topics:

Audience

This book is intended for chief technology officers, information technology architects, database administrators, system administrators, network administrators, and application administrators who perform the following tasks:

- Plan data centers
- Implement data center policies
- Maintain high availability systems
- Plan and build high availability solutions

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customer access to and use of Oracle support services will be pursuant to the terms and conditions specified in their Oracle order for the applicable services.

Related Documents

Knowledge of Oracle Database, Oracle RAC, and Data Guard concepts and terminology is required to understand the configuration and implementation details described in this book. For more information, see the Oracle Database documentation set. These books may be of particular interest:

- *Oracle Database Administrator's Guide*
- *Oracle Clusterware Administration and Deployment Guide*
- *Oracle Real Application Clusters Administration and Deployment Guide*
- *Oracle Automatic Storage Management Administrator's Guide*
- *Oracle Data Guard Concepts and Administration*
- *Oracle Database Backup and Recovery User's Guide*

Many books in the documentation set use the sample schemas of the seed database, which is installed by default when you install Oracle Database. See *Oracle Database Sample Schemas* for information about using these schemas.

Also, you can download the Oracle MAA best practice white papers at http://www.oracle.com/goto/maa.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Part I
Oracle Database High Availability Overview

1

Overview of High Availability

See the following topics to learn what high availability and why it is important. Then follow the roadmap to implementing a Maximum Availability Architecture.

What Is High Availability?

Availability is the degree to which an application and database service is available.

Availability is measured by the perception of an application's user. Users experience frustration when their data is unavailable or the computing system is not performing as expected, and they do not understand or care to differentiate between the complex components of an overall solution. Performance failures due to higher than expected usage create the same disruption as the failure of critical components in the architecture. If a user cannot access the application or database service, it is said to be unavailable. Generally, the term downtime is used to refer to periods when a system is unavailable.

Users who want their systems to be always ready to serve them need high availability. A system that is highly available is designed to provide uninterrupted computing services during essential time periods, during most hours of the day, and most days of the week throughout the year; this measurement is often shown as 24x365. Such systems may also need a high availability solution for planned maintenance operations such as upgrading a system's hardware or software.

Reliability, recoverability, timely error detection, and continuous operations are primary characteristics of a highly available solution:

- Reliability: Reliable hardware is one component of a high availability solution. Reliable software—including the database, web servers, and applications—is just as critical to implementing a highly available solution. A related characteristic is resilience. For example, low-cost commodity hardware, combined with software such as Oracle Real Application Clusters (Oracle RAC), can be used to implement a very reliable system. The resilience of an Oracle RAC database allows processing to continue even though individual servers may fail. For example, the Oracle RAC database allows processing to continue even though individual servers may fail.
- Recoverability: Even though there may be many ways to recover from a failure, it is important to determine what types of failures may occur in your high availability environment and how to recover from those failures quickly in order to meet your business requirements. For example, if a critical table is accidentally deleted from the database, what action should you take to recover it? Does your architecture provide the ability to recover in the time specified in a service-level agreement (SLA)?
- Timely error detection: If a component in your architecture fails, then fast detection is essential to recover from the unexpected failure. Although you may be able to recover quickly from an outage, if it takes an additional 90 minutes to discover the problem, then you may not meet your SLA. Monitoring the health of your environment requires reliable software to view it quickly and the ability to notify the database administrator of a problem.
- Continuous operation: Providing continuous access to your data is essential when very little or no downtime is acceptable to perform maintenance activities. Activities, such as moving a table to another location in the database or even adding CPUs to your hardware, should be transparent to the user in a high availability architecture.

More specifically, a high availability architecture should have the following traits:

- Tolerate failures such that processing continues with minimal or no interruption
- Be transparent to—or tolerant of—system, data, or application changes
- Provide built-in preventive measures
- Provide active monitoring and fast detection of failures
- Provide fast recoverability
- Automate detection and recovery operations
- Protect the data to minimize or prevent data loss and corruptions
- Implement the operational best practices to manage your environment
- Achieve the goals set in SLAs (for example, recovery time objectives (RTOs) and recovery point objectives (RPOs)) for the lowest possible total cost of ownership

Importance of Availability

The importance of high availability varies among applications. Databases and the internet have enabled worldwide collaboration and information sharing by extending the reach of database applications throughout organizations and communities.

This reach emphasizes the importance of high availability in data management solutions. Both small businesses and global enterprises have users all over the world who require access to data 24 hours a day. Without this data access, operations can stop, and revenue is lost. Users now demand service-level agreements from their information technology (IT) departments and solution providers, reflecting the increasing dependence on these solutions. Increasingly, availability is measured in dollars, euros, and yen, not just in time and convenience.

Enterprises have used their IT infrastructure to provide a competitive advantage, increase productivity, and empower users to make faster and more informed decisions. However, with these benefits has come an increasing dependence on that infrastructure. If a critical application becomes unavailable, then the business can be in jeopardy. The business might lose revenue, incur penalties, and receive bad publicity that has a lasting effect on customers and on the company's stock price.

It is important to examine the factors that determine how your data is protected and maximize availability to your users.

Cost of Downtime

The need to deliver increasing levels of availability continues to accelerate as enterprises reengineer their solutions to gain competitive advantage. Most often, these new solutions rely on immediate access to critical business data.

When data is not available, the operation can cease to function. Downtime can lead to lost productivity, lost revenue, damaged customer relationships, bad publicity, and lawsuits.

It is not always easy to place a direct cost on downtime. Angry customers, idle employees, and bad publicity are all costly, but not directly measured in currency. On the other hand, lost revenue and legal penalties incurred because SLA objectives are not met can easily be quantified. The cost of downtime can quickly grow in industries that are dependent on their solutions to provide service.

Other factors to consider in the cost of downtime are:

- The maximum tolerable length of a single unplanned outage

  If the event lasts less than 30 seconds, then it may cause very little impact and may be barely perceptible to users. As the length of the outage grows, the effect may grow exponentially and negatively affect the business.

- The maximum frequency of allowable incidents

  Frequent outages, even if short in duration, may similarly disrupt business operations.

When designing a solution, it is important to recognize the true cost of downtime to understand how the business can benefit from availability improvements.

Oracle provides a range of high availability solutions to fit every organization regardless of size. Small workgroups and global enterprises alike are able to extend the reach of their critical business applications. With Oracle and the Internet, applications and data are reliably accessible everywhere, at any time.

Causes of Downtime

One of the challenges in designing a high availability solution is examining and addressing all of the possible causes of downtime.

It is important to consider causes of both unplanned and planned downtime when designing a fault-tolerant and resilient IT infrastructure. Planned downtime can be just as disruptive to operations as unplanned downtime, especially in global enterprises that support users in multiple time zones.

The following table describes unplanned outage types and provides examples of each type.

Table 1-1 Causes of Unplanned Downtime

Type	Description	Examples
Site failure	A site failure may affect all processing at a data center, or a subset of applications supported by a data center. The definition of site varies given the contexts of on-premises and cloud. • Site failure - entire regional failure • Data center - entire data center location • Availability domain - isolated data center within a region with possibly many other availability domains • Fault domain - isolated set of system resources within an Availability Domain or data center Typically, each site, data center, availability domain, and fault domain has its own set of isolated hardware, DB compute, network, storage, and power.	• Extended site-wide power failure • Site-wide network failure • Natural disaster makes a data center inoperable • Terrorist or malicious attack on operations or the site

Table 1-1 (Cont.) Causes of Unplanned Downtime

Type	Description	Examples
Cluster-wide failure	The whole cluster hosting an Oracle RAC database is unavailable or fails. This includes: • Failures of nodes in the cluster • Failure of any other components that result in the cluster being unavailable and the Oracle database and instances on the site being unavailable	• The last surviving node on the Oracle RAC cluster fails and the node or database cannot be restarted • Both redundant cluster interconnections fail or Clusterware failure • Database corruption so severe that continuity is not possible on the current database server • Clusterware and hardware-software defects preventing availability or stability.
Computer failure	A computer failure outage occurs when the system running the database becomes unavailable because it has failed or is no longer available. When the database uses Oracle RAC then a computer failure represents a subset of the system (while retaining full access to the data).	• Database system hardware failure • Operating system failure • Oracle instance failure
Network failure	A network failure outage occurs when a network device stops or reduces network traffic and communication from your application to database, database to storage, or any system to system that is critical to your application service processing.	• Network switch failure • Network interface failure • Network cable failures
Storage failure	A storage failure outage occurs when the storage holding some or all of the database contents becomes unavailable because it has shut down or is no longer available.	• Disk or flash drive failure • Disk controller failure • Storage array failure

Table 1-1 (Cont.) Causes of Unplanned Downtime

Type	Description	Examples
Data corruption	A corrupt block is a block that was changed so that it differs from what Oracle Database expects to find. Block corruptions can be categorized as physical or logical: • In a physical block corruption, which is also called a media corruption, the database does not recognize the block at all; the checksum is invalid or the block contains all zeros. An example of a more sophisticated block corruption is when the block header and footer do not match. • In a logical block corruption, the contents of the block are physically sound and pass the physical block checks; however, the block can be logically inconsistent. Examples of logical block corruption include incorrect block type, incorrect data or redo block sequence number, corruption of a row piece or index entry, or data dictionary corruptions. Block corruptions can also be divided into interblock corruption and intrablock corruption: • In an intrablock corruption, the corruption occurs in the block itself and can be either a physical or a logical block corruption. • In an interblock corruption, the corruption occurs between blocks and can only be a logical block corruption. A data corruption outage occurs when a hardware, software, or network component causes corrupt data to be read or written. The service-level impact of a data corruption outage may vary, from a small portion of the application or database (down to a single database block) to a large portion of the application or database (making it essentially unusable).	• Operating system or storage device driver failure • Faulty host bus adapter • Disk controller failure • Volume manager error causing a bad disk read or write • Software or hardware defects
Human error	A human error outage occurs when unintentional or other actions are committed that cause data in the database to become incorrect or unusable. The service-level impact of a human error outage can vary significantly, depending on the amount and critical nature of the affected data.	• File deletion (at the file system level) • Dropped database object • Inadvertent data changes • Malicious data changes

Table 1-1 (Cont.) Causes of Unplanned Downtime

Type	Description	Examples
Lost or stray writes	A lost or stray write is another form of data corruption, but it is much more difficult to detect and repair quickly. A data block stray or lost write occurs when: • For a lost write, an I/O subsystem acknowledges the completion of the block write even though the write I/O did not occur in the persistent storage. On a subsequent block read on the primary database, the I/O subsystem returns the stale version of the data block, which might be used to update other blocks of the database, thereby corrupting it. • For a stray write, the write I/O completed but it was written somewhere else, and a subsequent read operation returns the stale value. • For an Oracle RAC system, a read I/O from one cluster node returns stale data after a write I/O is completed from another node (lost write). For example, this occurs if a network file system (NFS) is mounted in Oracle RAC without disabling attribute caching (for example, without using the `noac` option). In this case, the write I/O from one node is not immediately visible to another node because it is cached. Block corruptions caused by stray writes or lost writes can cause havoc to your database availability. The data block may be physically or logically correct but subsequent disk reads will show blocks that are stale or with an incorrect Oracle Database block address.	• Operating system or storage device driver failure • Faulty host bus adapter • Disk controller failure • Volume manager error • Other application software • Lack of network file systems (NFS) write visibility across a cluster • Software or hardware defects
Delay, slowdown, or hangs	A delay or slowdown occurs when the database or the application cannot process transactions because of a resource or lock contention. A perceived delay can be caused by lack of system resources.	• Database or application deadlocks • Runaway processes that consume system resources • Logon storms or system faults • Combination of application peaks with lack of system or database resources. This can occur with one application or many applications in a consolidated database environment without proper resource management. • Archived redo log destination or fast recovery area destination becomes full • Oversubscribed or heavily consolidated database system

The following table describes planned outage types and provides examples of each type.

Table 1-2 Causes of Planned Downtime

Type	Description	Examples
Software changes	• Planned periodic software changes to apply minor fixes for stability and security • Planned annual or bi-annual major upgrades to adopt new features and capabilities	• Software updates, including security updates to operating system, clusterware. or database • Major upgrade of operating system, clusterware, or database • Updating or upgrading application software
System and database changes	• Planned system changes to replace defected hardware • Planned system changes to expand or reduce system resources • Planned database changes to adopt parameter changes • Planned change to migrate to new hardware or architecture	• Adding or removing processors or memory to a server • Adding or removing nodes to or from a cluster • Adding or removing disks drives or storage arrays • Replacing any Field Replaceable Unit (FRU) • Changing configuration parameters • System platform migration • Migrating to cluster architecture • Migrating to new storage
Data changes	Planned data changes to the logical structure or physical organization of Oracle Database objects. The primary objective of these changes is to improve performance or manageability.	• Table definition changes • Adding table partitioning • Creating and rebuilding indexes
Application changes	Planned application changes can include data changes and schema and programmatic changes. The primary objective of these changes is to improve performance, manageability, and functionality.	Application upgrades

Oracle offers high availability solutions to help avoid both unplanned and planned downtime, and recover from failures. Oracle Database High Availability Solutions for Unplanned Downtime and Oracle Database High Availability Solutions for Planned Downtime discuss each of these high availability solutions in detail.

Chaos Engineering

Maximum Availability Architecture leverages Chaos Engineering throughout its testing and development life cycles to ensure that end-to-end application and database availability is preserved or at its optimal levels for any fault or maintenance event.

Chaos Engineering is the discipline of experimenting on a system in order to build confidence in the system's capability to withstand turbulent conditions in production. Specifically, MAA injects various faults and planned maintenance events to evaluate application and database impact throughout our development, stress, and testing cycles. With that experimentation, best practices, defects, and lessons learned are derived, and that knowledge is put back in practice to evolve and improve our MAA solutions.

Roadmap to Implementing the Maximum Availability Architecture

Oracle high availability solutions and sound operational practices are the key to successful implementation of an IT infrastructure. However, technology alone is not enough.

Choosing and implementing an architecture that best fits your availability requirements can be a daunting task. Oracle Maximum Availability Architecture (MAA) simplifies the process of choosing and implementing a high availability architecture to fit your business requirements with the following considerations:

- Encompasses redundancy across all components
- Provides protection and tolerance from computer failures, storage failures, human errors, data corruption, lost writes, system delays or slowdowns, and site disasters
- Recovers from outages as quickly and transparently as possible
- Provides solutions to eliminate or reduce planned downtime
- Provides consistent high performance and robust security
- Provides Oracle Engineered System and cloud options to simplify deployment and management and achieve the highest scalability, performance, and availability
- Achieves SLAs at the lowest possible total cost of ownership
- Applies to On-Premise, Oracle Public Cloud, and hybrid architectures consisting of parts on-premise and part in the cloud
- Provides special consideration to Container or Oracle Multitenant, Oracle Database In-Memory, and Oracle Sharding architectures

To build, implement, and maintain this type of architecture, you need to:

1. Analyze your specific high availability requirements, including both the technical and operational aspects of your IT systems and business processes, as described in High Availability and Data Protection – Getting From Requirements to Architecture.
2. Evaluate the various high availability architectures and their benefits and options, as described in Oracle MAA Reference Architectures.
3. Understand the availability impact for each MAA reference architecture, or various high availability features, on businesses and applications, as described in Oracle Database High Availability Solutions for Unplanned Downtime, and Oracle Database High Availability Solutions for Planned Downtime.
4. Familiarize yourself with Oracle high availability features, as described in Features for Maximizing Availability.
5. Use operational best practices to provide a successful MAA implementation, as described in Operational Prerequisites to Maximizing Availability .
6. Implement a high availability architecture using Oracle MAA resources, which provide technical details about the various Oracle MAA high availability technologies, along with best practice recommendations for configuring and using such technologies, such as Oracle MAA best practices white papers, customer papers with proof of concepts, customer case studies, recorded web casts, demonstrations, and presentations.

   Additional Oracle MAA resources are available at http://www.oracle.com/goto/maa.

2

High Availability and Data Protection – Getting From Requirements to Architecture

See the following topics to learn how Oracle Maximum Availability Architecture provides a framework to effectively evaluate the high availability requirements of an enterprise.

High Availability Requirements

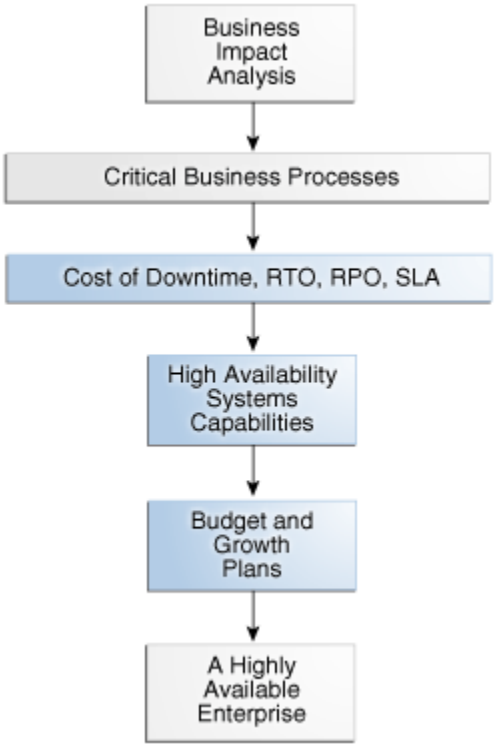
Any effort to design and implement a high availability strategy for Oracle Database begins by performing a thorough business impact analysis to identify the consequences to the enterprise of downtime and data loss, whether caused by unplanned or planned outages.

The term "business impact" is intended to be agnostic of whether the enterprise is a commercial venture, government agency, or not-for-profit institution. In all cases, data loss and downtime can seriously impact the ability of any enterprise to perform its functions. Implementing high availability may involve critical tasks such as:

- Retiring legacy systems
- Investing in more capable and robust systems and facilities
- Redesigning the overall IT architecture and operations to adapt to this high availability model
- Modifying existing applications to take full advantage of high availability infrastructures
- Redesigning business processes
- Hiring and training personnel
- Moving parts or an entire application or database into the Oracle Public Cloud
- Balancing the right level of consolidation, flexibility, and isolation
- Understanding the capabilities and limitations of your existing system and network infrastructure

By combining your business analysis with an understanding of the level of investment required to implement different high availability solutions, you can develop a high availability architecture that achieves both business and technical objectives.

Figure 2-1 Planning and Implementing a Highly Available Enterprise



A Methodology for Documenting High Availability Requirements

The elements of this analysis framework are:

- Business Impact Analysis
- Cost of Downtime
- Recovery Time Objective
- Recovery Point Objective
- Manageability Goal
- Total Cost of Ownership and Return on Investment

Business Impact Analysis

The business impact analysis categorizes the business processes based on the severity of the impact of IT-related outages.

A rigorous business impact analysis:

- Identifies the critical business processes in an organization
- Calculates the quantifiable loss risk for unplanned and planned IT outages affecting each of these business processes
- Outlines the effects of these outages
- Considers essential business functions, people and system resources, government regulations, and internal and external business dependencies

- Is based on objective and subjective data gathered from interviews with knowledgeable and experienced personnel
- Reviews business practice histories, financial reports, IT systems logs, and so on

For example, consider a semiconductor manufacturer with chip fabrication plants located worldwide. Semiconductor manufacturing is an intensely competitive business requiring a huge financial investment that is amortized over high production volumes. The human resource applications used by plant administration are unlikely to be considered as mission-critical as the applications that control the manufacturing process in the plant. Failure of the applications that support manufacturing affects production levels and have a direct impact on the financial results of the company.

As another example, an internal knowledge management system is likely to be considered mission-critical for a management consulting firm, because the business of a client-focused company is based on internal research accessibility for its consultants and knowledge workers. The cost of downtime of such a system is extremely high for this business.

Similarly, an e-commerce company is highly dependent on customer traffic to its website to generate revenue. Any disruption in service and loss of availability can dampen customer experience and drive away customers to the competition. Thus, the company needs to ensure that the existing infrastructure can scale and handle spikes in customer traffic. Sometimes, this is not possible using on-premise hardware and by moving the cloud the company can ensure their systems always remain operational.

Cost of Downtime

A complete business impact analysis provides the insight needed to quantify the cost of unplanned and planned downtime.

Understanding this cost is essential because it helps prioritize your high availability investment and directly influences the high availability technologies that you choose to minimize the downtime risk.

Various reports have been published, documenting the costs of downtime in different industries. Examples include costs that range from millions of dollars for each hour of brokerage operations and credit card sales, to tens of thousands of dollars for each hour of package shipping services.

These numbers are staggering. The Internet and Cloud can connect the business directly to millions of customers. Application downtime can disrupt this connection, cutting off a business from its customers. In addition to lost revenue, downtime can negatively affect customer relationships, competitive advantages, legal obligations, industry reputation, and shareholder confidence.

Recovery Time Objective

The business impact analysis determines your tolerance to downtime, also known as the recovery time objective (RTO).

An RTO is defined as the maximum amount of time that an IT-based business process can be down before the organization starts suffering unacceptable consequences (financial losses, customer dissatisfaction, reputation, and so on). RTO indicates the downtime tolerance of a business process or an organization in general.

RTO requirements are driven by the mission-critical nature of the business. Therefore, for a system running a stock exchange, the RTO is zero or near to zero.

An organization is likely to have varying RTO requirements across its various business processes. A high volume e-commerce website, for which there is an expectation of rapid response times, and for which customer switching costs are very low, the web-based customer interaction system that drives e-commerce sales is likely to have an RTO of zero or close to zero. However, the RTO of the systems that support back-end operations, such as shipping and billing, can be higher. If these back-end systems are down, then the business may resort to manual operations temporarily without a significant visible impact.

Some organizations have varying RTOs based on the probability of failures. One simple class separation is local failures (such as single database compute, disk/flash, network failure) as opposed to disasters (such as a complete cluster, database, data corruptions, or a site failure). Typically, business-critical customers have an RTO of less than 1 minute for local failures, and may have a higher RTO of less than 1 hour for disasters. For mission-critical applications the RTOs may indeed be the same for all unplanned outages.

Recovery Point Objective

The business impact analysis also determines your tolerance to data loss, also known as a recovery point objective (RPO).

The RPO is the maximum amount of data that an IT-based business process can lose without harm to the organization. RPO measures the data-loss tolerance of a business process or an organization in general. This data loss is often measured in terms of time, for example, zero, seconds, hours, or days of data loss.

A stock exchange where millions of dollars worth of transactions occur every minute cannot afford to lose any data. Therefore, its RPO must be zero. The web-based sales system in the e-commerce example does not require an RPO of zero, although a low RPO is essential for customer satisfaction. However, its back-end merchandising and inventory update system can have a higher RPO because lost data can be reentered.

An RPO of zero can be challenging for disasters, but I can be accomplished with various Oracle technologies protecting your database, especially Zero Data Loss Recovery Appliance.

Manageability Goal

A manageability goal is more subjective than either the RPO or the RTO. You must make an objective evaluation of the skill sets, management resources, and tools available in an organization, and the degree to which the organization can successfully manage all elements of a high availability architecture.

Just as RPO and RTO measure an organization's tolerance for downtime and data loss, your manageability goal measures the organization's tolerance for complexity in the IT environment. When less complexity is a requirement, simpler methods of achieving high availability are preferred over methods that may be more complex to manage, even if the latter could attain more aggressive RTO and RPO objectives. Understanding manageability goals helps organizations differentiate between what is possible and what is practical to implement.

Moving Oracle databases to Oracle Cloud can reduce manageability cost and complexity significantly, because Oracle Cloud lets you to choose between various Maximum Availability Architecture architectures with built-in configuration and life cycle operations. With Autonomous Database Cloud, database life cycle operations, such as backup and restore, software updates, and key repair operations are automatic.

Total Cost of Ownership and Return on Investment

Understanding the total cost of ownership (TCO) and objectives for return on investment (ROI) are essential to selecting a high availability architecture that also achieves the business goals of your organization.

TCO includes all costs (such as acquisition, implementation, systems, networks, facilities, staff, training, and support) over the useful life of your chosen high availability solution. Likewise, the ROI calculation captures all of the financial benefits that accrue for a given high availability architecture.

For example, consider a high availability architecture in which IT systems and storage at a remote standby site remain idle, with no other business use that can be served by the standby systems. The only return on investment for the standby site is the costs related to downtime avoided by its use in a failover scenario. Contrast this with a different high availability architecture that enables IT systems and storage at the standby site to be used productively while in the standby role (for example, for reports or for off-loading the overhead of user queries or distributing read-write workload from the primary system). The return on investment of such an architecture includes both the cost of downtime avoided and the financial benefits that accrue to its productive use, while also providing high availability and data protection.

Enterprises can also reduce TCO for growing infrastructure needs by moving workloads to the cloud rather than making an upfront capital investment in building a new data center. The major economic appeal is to convert capital expenditures into operational expenditures, and generate a higher ROI.

Mapping Requirements to Architectures

The business impact analysis will help you document what is already known. The outcome of the business impact analysis provides the insight you need to group databases having similar RTO and RPO objectives together.

Different applications, and the databases that support them, represent varying degrees of importance to the enterprise. A high level of investment in high availability infrastructure may not make sense for an application that if down, would not have an immediate impact on the enterprise. So where do you start?

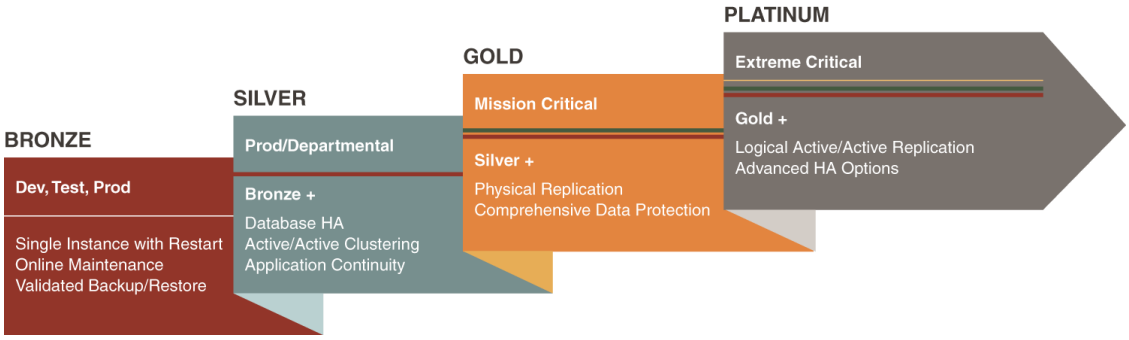
Groups of databases by similar RTO and RPO can be mapped to a controlled set of high availability reference architectures that most closely address the required service levels. Note that in the case where there are dependencies between databases, they are grouped with the database having the most stringent high availability requirement.

Oracle MAA Reference Architectures

Oracle MAA best practices define high availability reference architectures that address the complete range of availability and data protection required by enterprises of all sizes and lines of business.

The Platinum, Gold, Silver, and Bronze MAA reference architectures, or tiers, are applicable to on-premises, private and public cloud configurations, and hybrid cloud. They deliver the service levels described in the following figure.

Figure 2-2 Oracle MAA Reference Architectures



Each tier uses a different MAA reference architecture to deploy the optimal set of Oracle high availability capabilities that reliably achieve a given service level at the lowest cost and complexity. The tiers explicitly address all types of unplanned outages, including data corruption, component failure, and system and site outages, as well as planned outages due to maintenance, migrations, or other purposes.

Container databases (CDBs) using Oracle Multitenant can exist in any tier, Bronze through Platinum, providing higher consolidation density and higher TCO. Typically, the consolidation density is higher with Bronze and Silver tiers, and there is less or zero consolidation when deploying a Platinum tier.

Oracle Database In-Memory can also be leveraged in any of the MAA tiers. Because the In-Memory column store is seamlessly integrated into Oracle Database, all of the high availability benefits that come from the MAA tiers are inherited when implementing Oracle Database In-Memory.

Oracle Engineered Systems can also exist in any of the tiers. Integrating Zero Data Loss Recovery Appliance (Recovery Appliance) as the Oracle Database backup and recovery solution for your entire data center reduces RPO and RTO when restoring from backups. Leveraging Oracle Exadata Database Machine as your database platform in the MAA reference architectures provides the best database platform solution with the lowest RTO and brownout, along with additional Exadata MAA quality of service.

See Also:

High Availability Reference Architectures

Oracle Exadata Database Machine: Maximum Availability Architecture and MAA Best Practices for Oracle Exadata Database Machine

http://www.oracle.com/goto/maa for MAA white paper “Oracle Database In-Memory High Availability Best Practices”

Bronze Reference Architecture

The Bronze tier is appropriate for databases where simple restart of a failed component (e.g. listener, database instance, or database) or restore from backup is "HA and DR enough."

The Bronze reference architecture is based on a single instance Oracle Database using MAA best practices that implement the many capabilities for data protection and high availability

included with every Oracle Enterprise Edition license. Oracle-optimized backups using Oracle Recovery Manager (RMAN) provide data protection, and are used to restore availability should an outage prevent the database from restarting. The Bronze architecture then uses a redundant system infrastructure enhanced by Oracle's technologies, such as Oracle Restart, Recovery Manager (RMAN), Zero Data Loss Recovery Appliance, Flashback technologies, Online Redefinition, Online Patching, Automatic Storage Management (ASM), Oracle Multitenant, and more.

Silver Reference Architecture

The Silver tier provides an additional level of high availability for databases that require minimal or zero downtime in the event of database instance or server failure, as well as most common planned maintenance events, such as hardware and software updates.

The Silver reference architecture adds a rich set of enterprise capabilities and benefits, including clustering technology using either Oracle RAC or Oracle RAC One Node. Also, Application Continuity provides a reliable replay of in-flight transactions, which masks outages from users and simplifies application failover.

Gold Reference Architecture

The Gold tier raises the stakes substantially for business-critical applications that cannot tolerate high RTO and RPO for any disasters such as database, cluster, corruptions, or site failures. Additionally, major database upgrades or site migrations can be done in seconds.

The Gold tier also reduces costs while improving your return on investment by actively using all of the replicas at all times.

The Gold reference architecture adds database-aware replication technologies, Oracle Data Guard and Oracle Active Data Guard, which synchronize one or more replicas of the production database to provide real time data protection and availability. Database-aware replication substantially enhances high availability and data protection (corruption protection) beyond what is possible with storage replication technologies. Oracle Active Data Guard Far Sync is used for zero data loss protection at any distance.

See also, Oracle Data Guard Advantages Over Traditional Solutions.

Platinum Reference Architecture

The Platinum tier introduces several new Oracle Database capabilities, including Oracle GoldenGate for zero-downtime upgrades and migrations.

Edition Based Redefinition lets application developers design for zero-downtime application upgrades. You can alternativly design applications for Oracle Sharding, which provides extreme availability by distributing subsets of a database into highly available shards, while the application can access the entire database as one single logical database.

Each of these technologies requires additional effort to implement, but they deliver substantial value for the most critical applications where downtime is not an option.

High Availability and Data Protection Attributes by Tier

Each MAA reference architecture delivers known and tested levels of downtime and data protection.

The following table summarizes the high availability and data protection attributes inherent to each architecture. Each architecture includes all of the capabilities of the previous architecture,

and builds upon it to handle an expanded set of outages. The various components included and the service levels achieved by each architecture are described in other topics.

Table 2-1 High Availability and Data Protection Attributes By MAA Reference Architecture

MAA Reference Architecture	Unplanned Outages (Local Site)	Planned Maintenance	Data Protection	Unrecoverable Local Outages and Disaster Recovery
Bronze	Single Instance, auto-restart for recoverable instance and server failures. Redundancy for system infrastructure so that single component failures such as disk, flash, and network should not result in downtime.	Some online, most off-line	Basic runtime validation combined with manual checks	Restore from backup, potential to lose data generated since the last backup. Using Zero Data Loss Recovery Appliance reduces the potential to lose data to zero or near zero.
Silver	HA with automatic failover for instance and server failures	Most rolling, some online, few offline	Basic runtime validation combined with manual checks	Restore from backup, potential to lose data generated since the last backup. Using Zero Data Loss Recovery Appliance reduces the potential to lose data to zero or near zero. In-flight transactions are preserved with Application Continuity.
Gold	Comprehensive high availability and disaster recovery	All rolling or online	Comprehensive runtime validation combined with manual checks	Real-time failover, zero or near-zero data loss
Platinum	Zero application outage for Platinum ready applications	Zero application outage	Comprehensive runtime validation combined with manual checks	Zero application outage for Platinum-ready applications, with zero data loss. Oracle RAC, Oracle Active Data Guard, and Oracle GoldenGate complement each other, providing a wide array of solutions to achieve zero database service downtime for unplanned outages. Alternatively, use Oracle Sharding for site failure protection, because impact on the application is only on shards in failed site rather than the entire database. Each shard can be configured with real-time failover, zero or near-zero data loss, or zero application outage for Platinum-ready applications. In-flight transactions are preserved, with zero data loss.

> **See Also:**
>
> http://www.oracle.com/goto/maa

3

Features for Maximizing Availability

Familiarize yourself with the following Oracle Database high availability features used in MAA solutions.

Oracle Data Guard

Oracle Data Guard ensures high availability, data protection, and disaster recovery for enterprise data.

Data Guard provides a comprehensive set of services that create, maintain, manage, and monitor one or more standby databases to enable Oracle databases to survive outages of any kind, including natural disasters and data corruptions. A Data Guard standby database is an exact replica of the production database and thus can be transparently utilized in combination with traditional backup, restoration, flashback, and cluster techniques to provide the highest possible level of data protection, data availability and disaster recovery. Data Guard is included in Oracle Enterprise Edition.

A Data Guard configuration consists of one primary database and one or more standby databases. A primary database can be either a single-instance Oracle database or an Oracle RAC database. Similar to a primary database, a standby database can be either a single-instance Oracle database or an Oracle RAC database. Using a backup copy of the primary database, you can create up to 30 standby databases that receive redo directly from the primary database. Optionally you can use a cascaded standby to create Data Guard configurations where the primary transmits redo to a single remote destination, and that destination forwards redo to multiple standby databases. This enables a primary database to efficiently synchronize many more than 30 standby databases if desired.

Note:

Oracle Active Data Guard is an extension of basic Data Guard providing advanced features that off-load various types of processing from a production database, extend zero data loss protection over any distance, and that enhance high availability. Oracle Active Data Guard is licensed separately from Oracle Database Enterprise Edition.

There are several types of standby databases. Data Guard physical standby database is the MAA best practice for data protection and disaster recovery and is the most common type of standby database used. A physical standby database uses Redo Apply (an extension of Oracle media recovery) to maintain an exact, physical replica of the production database. When configured using MAA best practices, Redo Apply uses multiple Oracle-aware validation checks to prevent corruptions that can impact a primary database from impacting the standby. Other types of Data Guard standby databases include: snapshot standby (a standby open read/write for test or other purposes) and logical standby (used to reduce planned downtime).

Benefits of Using Data Guard

- Continuous Oracle-aware validation of all changes using multiple checks for physical and logical consistency of structures within an Oracle data block and redo, before updates are applied to a standby database. This isolates the standby database and prevents it from being impacted by data corruptions that can occur on the primary system.
- Transparent operation: There are no restrictions on the use of Data Guard physical standby for data protection. Redo Apply supports all data and storage types, all DDL

operations, and all applications (custom and packaged applications), and guarantees data consistency across primary and standby databases.

- Highest performance: Fast redo transport for best recovery point objective, fast apply performance for best recovery time objective. Multi-instance redo apply provides Oracle RAC scalability for redo apply, eliminating bottlenecks of a single database server. Redo apply can essentially scale up to available CPU, I/O, and network across your Oracle RAC cluster. An observed redo apply rate of 3500 MB per second (12 TB/hour) on 8 node RAC Exadata.
- Fast failover to a standby database to maintain availability should the primary database fail for any reason. Failover is either a manual or automatic operation depending on how Data Guard is configured.
- Integrated client notification framework to enable application clients to connect to a new primary database after a failover occurs.
- Automatic or automated (depending upon configuration) resynchronization of a failed primary database, quickly converting it to a synchronized standby database after a failover occurs.
- Choice of flexible data protection levels to support all network configurations, availability and performance SLAs, and business requirements.
- Management of a primary and all of its standby databases as a single configuration to simplify management and monitoring using either the Data Guard Broker command-line interface or Oracle Enterprise Manager Cloud Control.
- Data Guard Broker greatly improves manageability with additional features for comprehensive configuration health checks, resumable switchover operations, streamlined role transitions, support for cascaded standby configurations, and user-configurable thresholds for transport and apply lag to automatically monitor the ability of the configuration to support SLAs for recovery point and recovery time objectives at any instant in time.
- Efficient transport to multiple remote destinations using a single redo stream originating from the primary production database and forwarded by a cascading standby database.
- Snapshot Standby enables a physical standby database to be open read/write for testing or any activity that requires a read/write replica of production data. A snapshot standby continues to receive but does not apply updates generated by the primary. When testing is complete, a snapshot standby is converted back into a synchronized physical standby database by first discarding the changes made during the open read/write, and then applying the redo received from the primary database. Primary data is always protected. Snapshot standby is particularly useful when used in conjunction with Oracle Real Application Testing (workload is captured at the production database for replay and subsequent performance analysis at the standby database-an exact replica of production).
- Reduction of planned downtime by using a standby database to perform maintenance in a rolling manner. The only downtime is the time required to perform a Data Guard switchover; applications remain available while the maintenance is being performed.
- Increased flexibility for Data Guard configurations where the primary and standby systems may have different CPU architectures or operating systems subject to limitations defined in My Oracle Support note 413484.1.
- Efficient disaster recovery for a container database (CDB). Data Guard failover and switchover completes using a single command at a CDB level regardless of how many pluggable databases (PDBs) are consolidated within the CDB.
- Enables a specific administration privilege, SYSDG, to handle standard administration duties for Data Guard. This new privilege is based on the least privilege principle, in which

a user is granted only the necessary privileges required to perform a specific function and no more. The SYSDBA privilege continues to work as in previous releases.

- The Oracle Database In-Memory column store is supported on standby databases in an Active Data Guard environment.
- Further improves performance and availability of Data Warehouses in a Data Guard configuration by tracking information from `NOLOGGING` operations so they can be repaired with the new RMAN command `RECOVER DATABASE NOLOGGING`.
- Improves the impact multiple SYNC transport destinations have on the primary database through the use of a new parameter `DATA_GUARD_SYNC_LATENCY`. This parameter defines the maximum amount of time (in seconds) that the Primary database must wait before disconnecting subsequent destinations after at least one synchronous standby has acknowledged receipt of the redo.
- Data Guard Broker improves manageability by supporting destinations of different Endianess than the primary in addition to enhancing management of alternate destinations.
- Data Guard improves protection and Return To Operations (RTO) and Recovery Point Objectives (RPO) through multiple features including:
  - Multi-Instance Redo Apply (MIRA) provides scalable redo apply performance across Oracle RAC instances reducing RTO for even higher production OLTP or batch workloads
  - Compare primary and standby database blocks using the new `DBMS_DBCOMP` package to help identify lost writes so they can be resolved efficiently.
  - Fast Start Failover (FSFO) has the robustness of highly available zero data loss configurations with support for Maximum Protection mode while giving the flexibility of multiple observers and multiple failover targets for high availability in any configuration. FSFO can also be configured to automatically fail over to the standby with the detection of a lost write on the primary .
  - RPO is improved with no data loss failovers after a storage failure in ASYNC configurations and Data Guard Broker support for Application Continuity, improving the user experience during Data Guard role transitions.
- Oracle Data Guard Broker further improves the management of databases by supporting destinations of different endianness than the primary in addition to enhancing management of alternate archive destinations when the primary destination is unavailable.
- Oracle Data Guard Database Compare tool compares data blocks stored in an Oracle Data Guard primary database and its physical standby databases. Use this tool to find disk errors (such as lost write) that cannot be detected by other tools like the DBVERIFY utility. (new in Oracle Database 12c Release 2)
- Oracle Data Guard Broker supports multiple automatic failover targets in a fast-start failover configuration. Designating multiple failover targets significantly improves the likelihood that there is always a standby suitable for automatic failover when needed. (new in Oracle Database 12c Release 2)
- Dynamically change Oracle Data Guard Broker Fast-Start Failover target. The fast-start failover target standby database can be changed dynamically, to another standby database in the target list, without disabling fast-start failover. (new in Oracle Database 19c)
- Propagate restore points from primary to standby Site. Restore points created on the primary database are propagated to the standby sites, so that they are available even after a failover operation. (new in Oracle Database 19c)

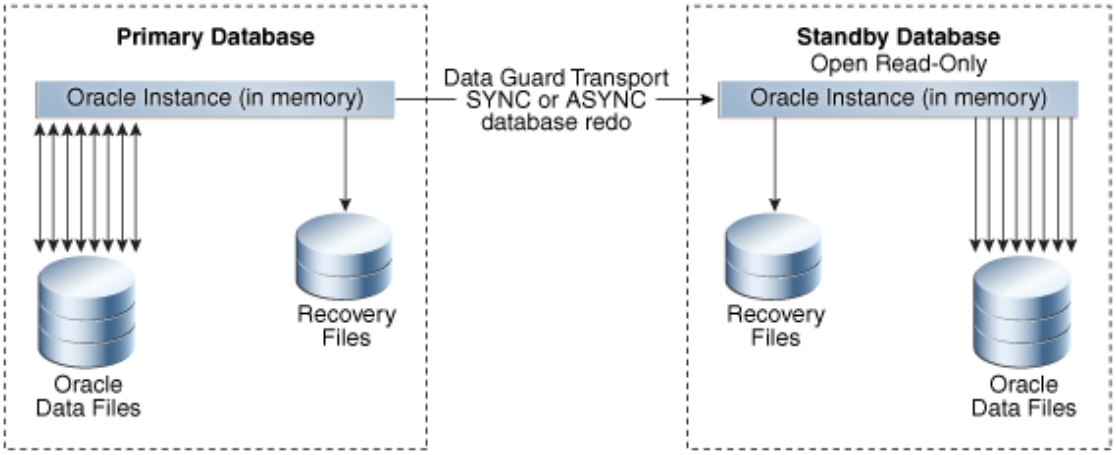
- Oracle Data Guard automatic outage resolution can be tuned to fit your specific needs. Oracle Data Guard has an internal mechanism to detect hung processes and terminate them, allowing the normal outage resolution to occur. (new in Oracle Database 19c)
- Active Data Guard DML redirection helps load balancing between the primary and standby databases. Incidental Data Manipulation Language (DML) operations can be run on Active Data Guard standby databases. This allows more applications to benefit from using an Active Data Guard standby database when some writes are required. When incidental DML is issued on an Active Data Guard standby database, the update is passed to the primary database where it is processed. The resulting redo of the transaction updates the standby database after which control is returned to the application. (new in Oracle Database 19c)

Oracle Active Data Guard

Oracle Active Data Guard is Oracle's strategic solution for real time data protection and disaster recovery for the Oracle database using a physical replication process.

Oracle Active Data Guard also provides high return on investment in disaster recovery systems by enabling a standby database to be open read-only while it applies changes received from the primary database. Oracle Active Data Guard is a separately licensed product that provides advanced features that greatly expand Data Guard capabilities included with Oracle Enterprise Edition.

Figure 3-1 Oracle Active Data Guard Architecture



Oracle Active Data Guard enables administrators to improve performance by offloading processing from the primary database to a physical standby database that is open read-only while it applies updates received from the primary database. Offload capabilities of Oracle Active Data Guard include read-only reporting and ad-hoc queries (including DML to global temporary tables and unique global or session sequences), data extracts, fast incremental backups, redo transport compression, efficient servicing of multiple remote destinations, and the ability to extend zero data loss protection to a remote standby database without impacting primary database performance. Oracle Active Data Guard also increases high availability by performing automatic block repair and enabling High Availability Upgrades automation.

Note:

Oracle Active Data Guard is licensed separately as a database option license for Oracle Database Enterprise Edition. All Oracle Active Data Guard capabilities are also included in an

Oracle Golden Gate license for Oracle Enterprise Edition. This provides customers with the choice of a standalone license for Oracle Active Data Guard, or licensing Oracle GoldenGate to acquire access to all advanced Oracle replication capabilities.

Benefits of Oracle Active Data Guard

Oracle Active Data Guard inherits all of the benefits previously listed for Data Guard, plus the following:

- Improves primary database performance: Production-offload to an Oracle Active Data Guard standby database of read-only applications, reporting, and ad hoc queries. Any application compatible with a read-only database can run on an Oracle Active Data Guard standby. Oracle also provides integration that enables the offloading of many Oracle E-Business Suite Reports, PeopleTools reporting, Oracle Business Intelligence Enterprise Edition (OBIEE), and Oracle TopLink applications to an Oracle Active Data Guard standby database.
- DML global temporary tables and the use of sequences at the standby database significantly expands the number of read-only applications that can be off-loaded from production databases to an Oracle Active Data Guard standby database.
- The unique ability to easily scale read performance using multiple Oracle Active Data Guard standby databases, also referred to as a Reader Farm.
- Production-offload of data extracts using Oracle Data Pump or other methods that read directly from the source database.
- Production-offload of the performance impact from network latency in a synchronous, zero data loss configuration where primary and standby databases are separated by hundreds or thousands of miles. Far sync uses a lightweight instance (control file and archive log files, but no recovery and no data files), deployed on a system independent of the primary database. The far sync instance is ideally located at the maximum distance from the primary system that an application can tolerate the performance impact of synchronous transport to provide optimal protection. Data Guard transmits redo synchronously to the far sync instance and far sync forwards the redo asynchronously to a remote standby database that is the ultimate failover target. If the primary database fails, the same failover command used for any Data Guard configuration, or mouse click using Oracle Enterprise Manager Cloud Control, or automatic failover using Data Guard Fast-Start Failover initiates a zero data loss failover to the remote destination. This transparently extends zero data loss protection to a remote standby database just as if it were receiving redo directly from the primary database, while avoiding the performance impact to the primary database of WAN network latency in a synchronous configuration.
- Production-offload of the overhead of servicing multiple remote standby destinations using far sync. In a far sync configuration, the primary database ships a single stream of redo to a far sync instance using synchronous or asynchronous transport. The far sync instance is able to forward redo asynchronously to as many as 29 remote destinations with zero incremental overhead on the source database.
- Data Guard maximum availability supports the use of the

  ```
  NOAFFIRM
  ```

  redo transport attribute. A standby database returns receipt acknowledgment to its primary database as soon as redo is received in memory. The standby database does not wait for the Remote File Server (RFS) to write to a standby redo log file.

  This feature provides increased primary database performance in Data Guard configurations using maximum availability and SYNC redo transport. Fast Sync isolates the primary database in a maximum availability configuration from any performance impact

due to slow I/O at a standby database. This new FAST SYNC feature can work with a physical standby target or within a far sync configuration.

- Production-offload of CPU cycles required to perform redo transport compression. Redo transport compression can be performed by the far sync instance if the Data Guard configuration is licensed for Oracle Advanced Compression. This conserves bandwidth with zero incremental overhead on the primary database.
- Production-offload and increased backup performance by moving fast incremental backups off of the primary database and to the standby database by utilizing Oracle Active Data Guard support for RMAN block change tracking.
- Increased high availability using Oracle Active Data Guard automatic block repair to repair block corruptions, including file header corruptions, detected at either the primary or standby, transparent to applications and users.
- Increased high availability by reducing planned downtime for upgrading to new Oracle Database patch sets and database releases using the additional automation provided by high availability Upgrade.
- Connection preservation on an Active Data Guard standby through a role change facilitates improved reporting and improves the user experience. The connections pause while the database role changes to a primary database and resume, improving the user experience.
- The Oracle Enterprise Manager Diagnostic tool can be used with Active Data Guard to capture and send performance data to the Automatic Workload Repository, while the SQL Tuning Advisor allows primary database SQL statement tuning to be offloaded to a standby database.
- Active Data Guard support for the Oracle Database In-Memory option enables reporting to be offloaded to the standby database while reaping the benefits the In-Memory option provides, including tailored column stores for the standby database workload.

Oracle Data Guard Advantages Over Traditional Solutions

Oracle Data Guard provides a number of advantages over traditional solutions.

- Fast, automatic or automated database failover for data corruptions, lost writes, and database and site failures, with recovery times of potentially seconds with Data Guard as opposed to hours with traditional solutions
- Zero data loss over wide area network using Oracle Active Data Guard Far Sync
- Offload processing for redo transport compression and redo transmission to up to 29 remote destinations using Oracle Active Data Guard Far Sync
- Automatic corruption repair automatically replaces a physical block corruption on the primary or physical standby by copying a good block from a physical standby or primary database
- Most comprehensive protection against data corruptions and lost writes on the primary database
- Reduced downtime for storage, Oracle ASM, Oracle RAC, system migrations and some platform migrations, and changes using Data Guard switchover
- Reduced downtime for database upgrades with Data Guard rolling upgrade capabilities
- Ability to off-load primary database activities—such as backups, queries, or reporting—without sacrificing the RTO and RPO ability to use the standby database as a read-only resource using the real-time query apply lag capability, including Database In-Memory column support

- Ability to integrate non-database files using Oracle Database File System (DBFS) or Oracle Advanced Cluster File System (Oracle ACFS) as part of the full site failover operations
- No need for instance restart, storage remastering, or application reconnection after site failures
- Transparency to applications
- Transparent and integrated support (application continuity and transaction guard) for application failover
- Effective network utilization
- Database In-Memory support
- Integrated service and client failover that reduces overall application RTO
- Enhanced and integrated Data Guard awareness with existing Oracle technologies such as Oracle RAC, RMAN, Oracle GoldenGate, Enterprise Manager, health check (`orachk`), DBCA, and Fleet Patch and Provisioning

For data resident in Oracle databases, Data Guard, with its built-in zero-data-loss capability, is more efficient, less expensive, and better optimized for data protection and disaster recovery than traditional remote mirroring solutions. Data Guard provides a compelling set of technical and business reasons that justify its adoption as the disaster recovery and data protection technology of choice, over traditional remote mirroring solutions.

Data Guard and Planned Maintenance

Data Guard standby databases can be used to reduce planned downtime by performing maintenance in a rolling fashion. Changes are implemented first at the standby database. The configuration is allowed to run with the primary at the old version and standby at the new version until there is confidence that the new version is ready for production. A Data Guard switchover can be performed, transitioning production to the new version or same changes can be applied to production in a rolling fashion. The only possible database downtime is the time required to perform the switchover.

There are several approaches to performing maintenance in a rolling fashion using a Data Guard standby. Customer requirements and preferences determine which approach is used.

Data Guard Redo Apply and Standby-First Patching

Beginning with Oracle Database 10g, there has been increased flexibility in cross-platform support using Data Guard Redo Apply.

In certain Data Guard configurations, primary and standby databases are able to run on systems having different operating systems (for example, Windows and Linux), word size (32bit/64bit), different storage, different Exadata hardware and software versions, or different hardware architectures. Redo Apply can also be used to migrate to Oracle Automatic Storage Management (ASM), to move from single instance Oracle databases to Oracle RAC, to perform technology refresh, or to move from one data center to the next.

Beginning with Oracle Database 11g Release 2 (11.2), Standby-First Patch Apply (physical standby using Redo Apply) can support different database software patch levels between a primary database and its physical standby database for the purpose of applying and validating Oracle patches in a rolling fashion. Patches eligible for Standby-First patching include:

- Database Release Updates (RUs) or Release Update Revisions (RURs)
- Database Patch Set Update (PSU)

- Database Critical Patch Update (CPU)
- Database bundled patch

Standby-First Patch Apply is supported for certified database software patches for Oracle Database Enterprise Edition 11g Release 2 (11.2) and later.

In each of the types of planned maintenance previously described, the configuration begins with a primary and physical standby database (in the case of migration to a new platform, or to ASM or Oracle RAC, the standby is created on the new platform). After all changes are implemented at the physical standby database, Redo Apply (physical replication) is used to synchronize the standby with the primary. A Data Guard switchover is used to transfer production to the standby (the new environment).

> **See Also:**
>
> My Oracle Support Note 413484.1 for information about mixed platform combinations supported in a Data Guard configuration.
>
> My Oracle Support Note 1265700.1 for more information about Standby First Patch Apply and the README for each patch to determine if a target patch is certified as being a Standby-First Patch.

Data Guard Transient Logical Rolling Upgrades

There are numerous types of maintenance tasks that are unable to use Redo Apply (physical replication) to synchronize the original version of a database with the changed or upgraded version. These tasks include:

- Database patches or upgrades that are not Standby-First Patch Apply-eligible. This includes database patch-sets (11.2.0.2 to 11.2.0.4) and upgrade to new Oracle Database releases (18c to 19c).
- Maintenance must be performed that modifies the physical structure of a database that would require downtime (for example, adding partitioning to non-partitioned tables, changing Basicfile LOBs to Securefile LOBs, changing XML-CLOB to Binary XML, or altering a table to be OLTP-compressed).

All of the previous types of maintenance can be performed in a rolling fashion using a Data Guard standby database by using Data Guard SQL Apply (logical replication) to synchronize the old and new versions of the database. Prior to Oracle Database 11g this required creating a logical standby database, performing the maintenance on the logical standby, resynchronizing the standby with the primary, and then switching over. Additionally if a physical standby was being used for disaster recovery, then a new physical standby database would have to be created from a backup of the production database at the new version. This represented a number of logistical and cost challenges when upgrading a multi-terabyte database.

Beginning with Oracle Database 11g, database rolling upgrades can use a new procedure called Transient Logical that begins and ends with a physical standby database. SQL Apply is only used during the phase when Data Guard is synchronizing across old and new versions. A new logical standby database does not need to be created if there is already a physical standby in place. A new physical standby database does not need to be created from a backup of the production database at the new version after the maintenance is complete. Similar to the traditional process of upgrading a Data Guard configuration having an in-place physical standby, the original primary is upgraded or changed using redo from the new primary

database and Redo Apply (a single catalog upgrade migrates both primary and standby databases to the new Oracle release).

Transient Logical upgrades require that the primary database be at Oracle Database 11g release 1 (11.1) or later and that the database meet the prerequisites of SQL Apply.

Oracle provides a Bourne shell script that automates a number of the manual steps required by the Transient Logical rolling upgrade process.

Databases that use Oracle Database Vault can be upgraded to new Oracle Database releases and patch sets by using Oracle Data Guard database rolling upgrades (transient logical standby only).

> **See Also:**
>
> http://www.oracle.com/goto/maa for Oracle MAA white paper “Oracle Database Rolling Upgrades: Using a Data Guard Physical Standby Database”

Rolling Upgrade Using Oracle Active Data Guard

Rolling database upgrade using Oracle Active Data Guard provides a simpler, automated, and easily repeatable method for reducing planned downtime than represented by the manual Transient Logical rolling upgrade procedure.

Rolling upgrade using Oracle Active Data Guard transforms the 42 or more steps required by the manual procedure into several easy-to-use DBMS_ROLLING PL/SQL packages. Rolling upgrades performed using the DBMS_ROLLING PL/SQL package are supported on a multitenant container database (CDB).

A rolling upgrade using Oracle Active Data Guard:

- Generates an upgrade plan with a configuration-specific set of instructions to guide you through the upgrade process.
- Modifies parameters of the rolling upgrade.
- Configures primary and standby databases participating in the upgrade.
- Performs switchover of the production database to the new version. Switchover is the only downtime required.
- Completes the upgrade of the old primary and any additional standby databases in the Data Guard configuration and resynchronizes with the new primary.

Rolling upgrade using Oracle Active Data Guard has the following additional benefits:

- Provides a simple specify-compile-run protocol
  - Catches configuration errors at the compilation step
  - Runtime errors are detected during processing
- The state is kept in the database
  - Enables a reliable, repeatable process
- Runtime steps are constant regardless of how many databases are involved
- Handles failure at the original primary database
- Enables data protection for the upgraded primary at all times

> **See Also:**
>
> http://www.oracle.com/goto/maa for Oracle MAA white paper “Oracle Database Rolling Upgrades: Using a Data Guard Physical Standby Database”
>
> *Oracle Data Guard Concepts and Administration*

Oracle GoldenGate

Oracle GoldenGate is Oracle's strategic logical replication solution for data distribution and data integration.

Oracle GoldenGate offers a real-time, log-based change data capture and replication software platform. The software provides capture, routing, transformation, and delivery of transactional data across heterogeneous databases in real time.

Unlike replication solutions from other vendors, Oracle GoldenGate is more closely integrated with Oracle Database while also providing an open, modular architecture ideal for replication across heterogeneous database management systems. This combination of attributes eliminates compromise, making Oracle GoldenGate the preferred logical replication solution for addressing requirements that span Oracle Database and non-Oracle Database environments.

A typical environment includes a capture, pump, and delivery process. Each of these processes can run on most of the popular operating systems and databases, including Oracle Database. All or a portion of the data can be replicated, and the data within any of these processes can be manipulated for not only heterogeneous environments but also different database schemas, table names, or table structures. Oracle GoldenGate also supports bidirectional replication with preconfigured conflict detection and resolution handlers to aid in resolving data conflicts.

Oracle GoldenGate logical replication enables all databases in an Oracle GoldenGate configuration, both source and target databases, to be open read-write. This makes it a key component of MAA for addressing a broad range of high availability challenges for zero downtime maintenance, cross platform migration, and continuous data availability, specifically:

- **Zero or near zero downtime maintenance.** In this architecture, Oracle GoldenGate provides greater flexibility than the capabilities provided by Data Guard. Oracle GoldenGate source and target databases can have a different physical and logical structure, can reside on different hardware and operating system architectures, can span wide differences in Oracle Database releases (for example, 12.2 to 19c), or be a mix of Oracle and non-Oracle systems. This allows for the modernization of 24x7 servers and allows new Oracle features to be implemented without impacting the availability of the databases. Maintenance is first performed on a target database while production runs on the source. After the maintenance is complete, production can be moved to the source all at once, similar to a Data Guard switchover. Optionally, bidirectional replication can be used to gradually move users over to the new system to create the perception of zero downtime. In either case, Oracle GoldenGate replication can be enabled in the reverse direction to keep the original source database synchronized during a transition period, making it simple to effect a planned fall-back to the previous version if needed, with minimal downtime and no data loss.
- **Zero or near-zero downtime migrations when a Data Guard solution is not applicable.** Platform or database migrations can be carried out using Oracle GoldenGate as the data synchronization method between the old and new systems. Once the database has been instantiated on another host, Oracle GoldenGate is configured to replicate

changes from the production database. A guaranteed restore point can be created on the migrated database so that after user testing the database can be flashed back, and Oracle GoldenGate can apply any outstanding data changes from the production database before moving the application users to the new database, similar to a snapshot standby database. If desired, bi-directional replication can also be configured from the migrated database back to the production database for use as a fallback solution.

- **Zero or near-zero downtime application upgrades.** Application upgrades that modify back-end database objects typically result in significant planned downtime while maintenance is being performed. Oracle GoldenGate replication enables data transformations that map database objects used by a previous version of an application to objects modified by the new version of an application. This enables database maintenance to be performed on a separate copy of the production database without impacting the availability of the application. After the maintenance is complete and Oracle GoldenGate has finished synchronizing old and new versions, users can be switched to the new version of the application.
- **Read-write access to a replica database while it is being synchronized with its source database.** This is most often used to offload reporting to a copy of a production database when the reporting application requires a read-write connection to database in order to function. This is also relevant to disaster recovery environments where the nature of the technology used for the application tier requires an active read-write connection to the DR database at all times in order to meet recovery time objectives.
- **Active-Active replication.** Oracle GoldenGate supports an active-active multi-directional configuration, where there are two or more systems with identical sets of data that can be changed by application users on either system. Oracle GoldenGate replicates transactional data changes from each database to the others to keep all sets of data current.
- Seamless moves between Oracle Real Application Clusters (RAC) nodes in the event of database instance failure or during applicable maintenance operations. This ability provides high availability with Oracle GoldenGate and it is possible to patch and upgrade the Oracle GoldenGate software on one or more nodes in the cluster without affecting the node where Oracle GoldenGate is currently running. Then at a predetermined time, Oracle GoldenGate can be switched to one of the upgraded nodes. The switch is done without reconfiguring Oracle GoldenGate because configuration information is shared across the Oracle RAC cluster.

> **See Also:**
>
> Oracle GoldenGate Documentation
>
> http://www.oracle.com/goto/maa for Oracle MAA Oracle GoldenGate white papers

Best Practice: Oracle Active Data Guard and Oracle GoldenGate

While Oracle Active Data Guard and Oracle GoldenGate are each capable of maintaining a synchronized copy of an Oracle database, each has unique characteristics that result in high availability architectures that can use one technology or the other, or both at the same time, depending upon requirements.

Examples of MAA Best Practice guidelines are as follows:

When to Use Oracle Active Data Guard

Use Oracle Active Data Guard when the emphasis is on simplicity, data protection, and availability.

- Simplest, fastest, one-way replication of a complete Oracle database.
- No restrictions: Data Guard Redo Apply supports all data and storage types and Oracle features; transparent replication of DDL
- Features optimized for data protection: Detects silent corruptions that can occur on source or target; automatically repairs corrupt blocks
- Synchronized standby open read-only provides simple read-only offloading for maximum ROI
- Transparency of backups: A Data Guard primary and standby are physically exact copies of each other; RMAN backups are completely interchangeable
- Zero data loss protection at any distance, without impacting database performance
- Minimizing planned downtime and risk using standby first patching, database rolling upgrades, and select platform migrations
- Reduce risk of introducing change by dual purposing a DR system for testing using Data Guard Snapshot Standby
- Integrated automatic database and client failover
- Integrated management of a complete configuration: Data Guard Broker command line interface or Oracle Enterprise Manager Cloud Control

When to Use Oracle GoldenGate

Use Oracle GoldenGate when the emphasis is on advanced replication requirements not addressed by Oracle Active Data Guard.

- Any requirement where the replica database must be open read/write while synchronizing with the primary database
- Any data replication requirements such as multimaster and bidirectional replication, subset replication, many-to-one replication, and data transformations.
- When data replication is required between endian format platforms or across-database major versions.
- Maintenance and migrations where zero downtime or near zero downtime is required. Oracle GoldenGate can be used to migrate between application versions, for example, from Application 1.0 to Application 2.0 without downtime.
- Database rolling upgrades where it is desired to replicate from new version down to the old version for the purpose of fast fall-back if something is wrong with the upgrade.
- Zero downtime planned maintenance where bidirectional replication is used to gradually migrate users to the new version, creating the perception of zero downtime. Note that bidirectional replication requires avoiding or resolving update conflicts that can occur on disparate databases.

When to Use Oracle Active Data Guard and Oracle GoldenGate Together

Oracle Active Data Guard and Oracle GoldenGate are not mutually exclusive. The following are use cases of high availability architectures that include the simultaneous use of Oracle Active Data Guard and Oracle GoldenGate.

- An Oracle Active Data Guard standby is utilized for disaster protection and database rolling upgrades for a mission critical OLTP database. At the same time, Oracle GoldenGate is used to replicate data from the Data Guard primary database (or from the standby database using Oracle GoldenGate ALO mode) for ETL update of an enterprise data warehouse.
- Oracle GoldenGate subset replication is used to create an operational data store (ODS) that extracts, transforms, and aggregates data from numerous data sources. The ODS supports mission critical application systems that generate significant revenue for the company. An Oracle Active Data Guard standby database is used to protect the ODS, providing optimal data protection and availability.
- Oracle GoldenGate bidirectional replication is utilized to synchronize two databases separated by thousands of miles. User workload is distributed across each database based upon geography, workload, and service level using Global Data Services (GDS). Each Oracle GoldenGate copy has its own local synchronous Data Guard standby database that enables zero data loss failover if an outage occurs. Oracle GoldenGate capture and apply processes are easily restarted on the new primary database following a failover because the primary and standby are an exact, up-to-date replica of each other.
- An Oracle Active Data Guard standby database used for disaster protection is temporarily converted into an Oracle GoldenGate target for the purpose of performing planned maintenance not supported by Data Guard. For example, a Siebel application upgrade requiring modification of back-end database objects which require comprehensive testing before switching users over to the new system.
- Oracle Active Data Guard is used to protect a production environment when a major database version upgrade is required offering zero or near-zero downtime (for example, Oracle 18c to 19c.) A second primary/standby environment is created using the new database version, and Oracle GoldenGate is used to replicate data from the production environment to the copy with one-way or bidirectional replication. When Oracle GoldenGate has completed synchronizing the old and new environments, production is switched to the new environment and the old environment is decommissioned. This provides zero or minimal downtime depending upon configuration, eliminates risk by providing complete isolation between the old and new environment, and avoids any impact to data protection and availability SLAs if problems are encountered during the upgrade process.

> **See Also:**
>
> http://www.oracle.com/goto/maa for Oracle MAA Best Practices white paper ““Transparent Role Transitions With Oracle Data Guard and Oracle GoldenGate"

Recovery Manager

Recovery Manager (RMAN) provides a comprehensive foundation for efficiently backing up and recovering the database. RMAN eliminates operational complexity while providing superior performance and availability of the database.

RMAN determines the most efficient method of running the requested backup, restoration, or recovery operation and then submits these operations to the Oracle Database server for processing. RMAN and the server automatically identify modifications to the structure of the database and dynamically adjust the required operation to adapt to the changes.

RMAN is the standard interface to backup and restore from Recovery Appliance, local disk (ZFS storage), tape, and cloud object store.

RMAN provides the following benefits:

- Support for Oracle Sharding - RMAN support for every independent database (shard)
- Enhancement for Sparse Databases - allows backup and restore to operate on

  `SPARSE`

  backup sets and or image copies
- Over the Network Standby Database repair of

  `NONLOGGED`

  operation - new syntax for validation and repair on Standby -

  ```
  VALIDATE/RECOVER .. NONLOGGED BLOCK;
  ```

- `RMAN DUPLICATE`

  feature enhanced to support creation of Far Sync from Primary and backup
- `RMAN DUPLICATE`

  Using Encrypted Backups - RMAN enhanced support non Auto-login wallet based encrypted backups with a new

  `SET`

  command - enables interrupt-free cloning
- Support for cross-platform backup and restore over the network
- Network-enabled restoration allows the

  `RESTORE`

  operations to copy data files directly from one database to another over the network
- Simplified table restoration with the

  `RECOVER TABLE`

  command
- Support for Oracle Multitenant, including backup and recovery of individual pluggable databases

- Support for cross-platform Oracle Multitenant, including backup and recovery of individual PDBs
- Automatic channel failover on backup and restore operations
- Automatic failover to a previous backup when the restore operation discovers a missing or corrupt backup
- Automatic creation of new database files and temporary files during recovery
- Automatic recovery through a previous point-in-time recovery—recovery through reset logs
- Block media recovery, which enables the data file to remain online while fixing the block corruption
- Fast incremental backups using block change tracking
- Fast backup and restore operations with intrafile and interfile parallelism
- Enhanced security with a virtual private recovery catalog
- Merger of incremental backups into image copies, providing up-to-date recoverability
- Optimized backup and restoration of required files only
- Retention policy to ensure that relevant backups are retained
- Ability to resume backup and restore operations in case of failure
- Automatic backup of the control file and the server parameter file, ensuring that backup metadata is available in times of database structural changes and media failure and disasters
- Easily reinstantiate a new database from an existing backup or directly from the production database (thus eliminating staging areas) using the

  `DUPLICATE`

  command.

> **See Also:**
>
> *Oracle Database Backup and Recovery User's Guide*

Oracle Real Application Clusters and Oracle Clusterware

Oracle RAC and Oracle Clusterware enable Oracle Database to run any packaged or custom application across a set of clustered servers.

This capability provides the highest levels of availability and the most flexible scalability. If a clustered server fails, then Oracle Database continues running on the surviving servers. When more processing power is needed, you can add another server without interrupting access to data.

Oracle RAC enables multiple instances that are linked by an interconnect to share access to an Oracle database. In an Oracle RAC environment, Oracle Database runs on two or more systems in a cluster while concurrently accessing a single shared database. The result is a single database system that spans multiple hardware systems, enabling Oracle RAC to provide high availability and redundancy during failures in the cluster. Oracle RAC

accommodates all system types, from read-only data warehouse systems to update-intensive online transaction processing (OLTP) systems.

Oracle Clusterware is software that, when installed on servers running the same operating system, enables the servers to be bound together to operate as if they are one server, and manages the availability of user applications and Oracle databases. Oracle Clusterware also provides all of the features required for cluster management, including node membership, group services, global resource management, and high availability functions:

- For high availability, you can place Oracle databases (single-instance or Oracle RAC databases), and user applications (Oracle and non-Oracle) under the management and protection of Oracle Clusterware so that the databases and applications restart when a process fails or so that a failover to another node occurs after a node failure.
- For cluster management, Oracle Clusterware presents multiple independent servers as if they are a single-system image or one virtual server. This single virtual server is preserved across the cluster for all management operations, enabling administrators to perform installations, configurations, backups, upgrades, and monitoring functions. Then, Oracle Clusterware automatically distributes the processing of these management functions to the appropriate nodes in the cluster.

Oracle Clusterware is a requirement for using Oracle RAC. Oracle Clusterware is the only clusterware that you need for most platforms on which Oracle RAC operates. Although Oracle Database continues to support third-party clusterware products on specified platforms, using Oracle Clusterware provides these main benefits:

- Dispenses with proprietary vendor clusterware
- Uses an integrated software stack from Oracle that provides disk management with local or remote Oracle Automatic Storage Management (Oracle Flex ASM) to data management with Oracle Database and Oracle RAC
- Can be configured in large clusters, called an Oracle Flex Cluster.

In addition, Oracle Database features, such as Oracle services, use the underlying Oracle Clusterware mechanisms to provide their capabilities.

Oracle Clusterware requires two clusterware components: a voting disk to record node membership information and the Oracle Cluster Registry (OCR) to record cluster configuration information. The voting disk and the OCR must reside on shared storage. Oracle Clusterware requires that each node be connected to a private network over a private interconnect.

Benefits of Using Oracle Clusterware

Oracle Clusterware provides the following benefits.

- Tolerates and quickly recovers from computer and instance failures.
- Simplifies management and support by means of using Oracle Clusterware together with Oracle Database. By using fewer vendors and an all Oracle stack you gain better integration compared to using third-party clusterware.
- Performs rolling upgrades for system and hardware changes. For example, you can apply Oracle Clusterware upgrades, patch sets, and interim patches in a rolling fashion.

  When you upgrade to Oracle Database 12c, Oracle Clusterware and Oracle ASM binaries are installed as a single binary called the Oracle Grid Infrastructure. You can upgrade Oracle Clusterware in a rolling manner from Oracle Clusterware 10g and Oracle Clusterware 11g; however, you can only upgrade Oracle ASM in a rolling manner from Oracle Database 11g release 1 (11.1).
- Automatically restarts failed Oracle processes.

- Automatically manages the virtual IP (VIP) address. When a node fails, the node's VIP address fails over to another node on which the VIP address can accept connections.
- Automatically restarts resources from failed nodes on surviving nodes.
- Controls Oracle processes as follows:
  - For Oracle RAC databases, Oracle Clusterware controls all Oracle processes by default.
  - For Oracle single-instance databases, Oracle Clusterware enables you to configure the Oracle processes into a resource group that is under the control of Oracle Clusterware.
- Provides an application programming interface (API) for Oracle and non-Oracle applications that enables you to control other Oracle processes with Oracle Clusterware, such as restart or react to failures and certain rules.
- Manages node membership and prevents split-brain syndrome in which two or more instances attempt to control the database.
- Using server weight-based node eviction allows for aligning the choice of which node gets evicted in case of certain failures in the cluster with business requirements, ensuring that the most important workload is kept alive for as long as possible, assuming an equal choice between servers.
- Provides the ability to perform rolling release upgrades of Oracle Clusterware, with no downtime for applications.

Benefits of Using Oracle Real Application Clusters and Oracle Clusterware

Together, Oracle RAC and Oracle Clusterware provide all of the Oracle Clusterware benefits plus the following benefits.

- Provides better integration and support of Oracle Database by using an all Oracle software stack compared to using third-party clusterware.
- Relocate Oracle Service automatically. Plus, when you perform additional fast application notification (FAN) and client configuration, distribute FAN events so that applications can react immediately to achieve fast, automatic, and intelligent connection and failover.
- Detect connection failures fast and automatically, and remove terminated connections for any Java application using Oracle Universal Connection Pool (Oracle UCP) Fast Connection Failover and FAN events.
- Balance work requests using Oracle UCP runtime connection load balancing.
- Use runtime connection load balancing with Oracle UCP, Oracle Call Interface (OCI), and Oracle Data Provider for .NET (ODP.NET).
- Distribute work across all available instances using load balancing advisory.
- You can configure a database so that Oracle Clusterware is aware of the CPU requirements and limits for the given database. Oracle Clusterware uses this information to place the database resource only on servers that have a sufficient number of CPUs, amount of memory, or both.
- Allow the flexibility to increase processing capacity using commodity hardware without downtime or changes to the application.
- Provide comprehensive manageability integrating database and cluster features.
- Provide scalability across database instances.
- Implement Fast Connection Failover for nonpooled connections.

Oracle RAC Advantages Over Traditional Cold Cluster Solutions

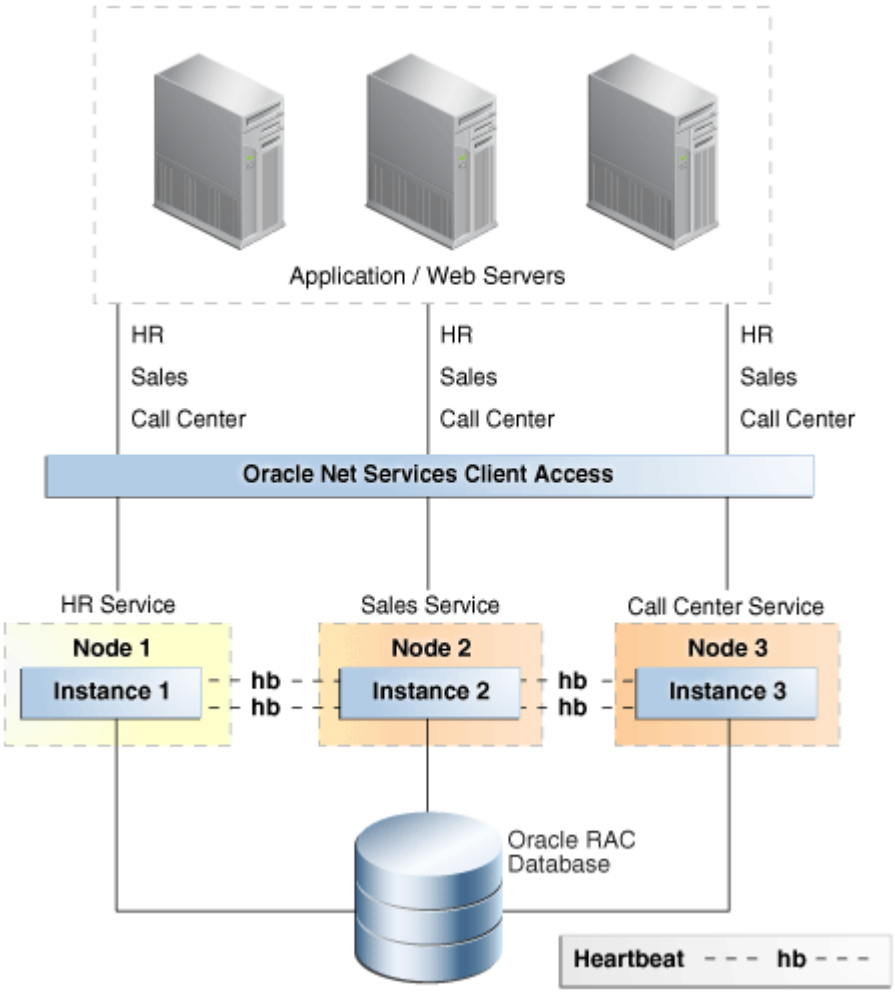
Oracle RAC provides many advantages over traditional cold cluster solutions, including the following.

- Scalability across database instances
- Flexibility to increase processing capacity using commodity hardware without downtime or changes to the application
- Ability to tolerate and quickly recover from computer and instance failures (measured in seconds)
- Application brownout can be zero or seconds compared to minutes and hours with cold cluster solutions
- Optimized communication in the cluster over redundant network interfaces, without using bonding or other technologies

  Oracle Grid Infrastructure and Oracle RAC make use of Redundant Interconnect Usage that distributes network traffic and ensures optimal communication in the cluster. This functionality is available starting with Oracle Database 11g Release 2 (11.2.0.2). In previous releases, technologies like bonding or trunking were used to make use of redundant networks for the interconnect.
- Rolling upgrades for system and hardware changes
- Rolling patch upgrades for some interim patches, security patches, CPUs, and cluster software
- Fast, automatic, and intelligent connection and service relocation and failover
- Comprehensive manageability integrating database and cluster features with Grid Plug and Play and policy-based cluster and capacity management
- Load balancing advisory and run-time connection load balancing help redirect and balance work across the appropriate resources
- Oracle Quality of Service (QoS) Management for policy-based run-time management of resource allocation to database workloads to ensure service levels are met in order of business need under dynamic conditions. This is accomplished by assigning a service to a server pool where the database is running. Resources from the pool are used to make sure the required capacity is available.
- Oracle Enterprise Management support for Oracle Automatic Storage Management (ASM) and Oracle Advanced Cluster File System (ACFS), Grid Plug and Play, Cluster Resource Management, Oracle Clusterware and Oracle RAC Provisioning and patching.
- SCAN (Single Client Access Name) support as a single name to the clients connecting to Oracle RAC that does not change throughout the life of the cluster, even if you add or remove nodes from the cluster.

The following figure shows Oracle Database with Oracle RAC architecture. This figure shows Oracle Database with Oracle RAC architecture for a partitioned three-node database. An Oracle RAC database is connected to three instances on different nodes. Each instance is associated with a service: HR, Sales, and Call Center. The instances monitor each other by checking "heartbeats." Oracle Net Services provide client access to the Application/web server tier at the top of the figure.

Figure 3-2 Oracle Database with Oracle RAC Architecture



Note:

After Oracle release 11.2, Oracle RAC One Node or Oracle RAC is the preferred solution over Oracle Clusterware (Cold Cluster Failover) because it is a more complete and feature-rich solution.

See Also:

Oracle RAC Administration and Deployment Guide

Oracle Clusterware Administration and Deployment Guide

Oracle RAC One Node

Oracle Real Application Clusters One Node (Oracle RAC One Node) is a single instance of an Oracle RAC database that runs on one node in a cluster.

This feature enables you to consolidate many databases into one cluster with minimal overhead, protecting them from both planned and unplanned downtime. The consolidated databases reap the high availability benefits of failover protection, online rolling patch application, and rolling upgrades for the operating system and Oracle Clusterware.

Oracle RAC One Node enables better availability than cold failover for single-instance databases because of the Oracle technology called online database relocation, which intelligently migrates database instances and connections to other cluster nodes for high availability and load balancing. Online database relocation is performed using the Server Control Utility (SRVCTL).

Oracle RAC One Node provides the following:

- Always available single-instance database services
- Built-in cluster failover for high availability
- Live migration of instances across servers
- Online rolling patches and rolling upgrades for single-instance databases
- Online upgrade from single-instance to multiple-instance Oracle RAC
- Better consolidation for database servers
- Enhanced server virtualization
- Lower cost development and test platform for full Oracle RAC
- Relocation of Oracle RAC primary and standby databases configured with Data Guard. This functionality is available starting with Oracle Database 11g Release 2 (11.2.0.2).

Oracle RAC One Node also facilitates the consolidation of database storage, standardizes your database environment, and, when necessary, enables you to transition to a full, multiple-instance Oracle RAC database without downtime or disruption.

Oracle Automatic Storage Management

Oracle Automatic Storage Management (Oracle ASM) provides a vertically integrated file system and volume manager directly in the Oracle Database kernel.

This design provides several benefits, resulting in:

- Significantly less work to provision database storage
- Higher level of availability
- Elimination of the expense, installation, and maintenance of specialized storage products
- Unique capabilities for database applications

For optimal performance, Oracle ASM spreads files across all available storage. To protect against data loss, Oracle ASM extends the concept of SAME (stripe and mirror everything) and adds more flexibility because it can mirror at the database file level rather than at the entire disk level.

More important, Oracle ASM simplifies the processes of setting up mirroring, adding disks, and removing disks. Instead of managing hundreds or possibly thousands of files (as in a large data warehouse), database administrators using Oracle ASM create and administer a larger-grained object called a disk group. The disk group identifies the set of disks that are managed as a logical unit. Automation of file naming and placement of the underlying database files save administrators time and ensure adherence to standard best practices.

The Oracle ASM native mirroring mechanism (two-way or three-way) protects against storage failures. With Oracle ASM mirroring, you can provide an additional level of data protection with the use of failure groups. A failure group is a set of disks sharing a common resource (disk controller or an entire disk array) whose failure can be tolerated. After it is defined, an Oracle ASM failure group intelligently places redundant copies of the data in separate failure groups. This ensures that the data is available and transparently protected against the failure of any component in the storage subsystem.

By using Oracle ASM, you can:

- Mirror and stripe across drives and storage arrays.
- Automatically remirror from a failed drive to remaining drives.
- Automatically rebalance stored data when disks are added or removed while the database remains online.
- Support Oracle database files and non-database files using Oracle Advanced Cluster File System (Oracle ACFS).
- Allow for operational simplicity in managing database storage.
- Manage the Oracle Cluster Registry (OCR) and voting disks.
- Provide preferred read capability on disks that are local to the instance, which gives better performance for an extended cluster.
- Support very large databases.
- Support Oracle ASM rolling upgrades.
- Improve availability and reliability using the Oracle ASM disk scrubbing process to find and repair logical data corruptions using mirror disks.
- Support finer granularity in tuning and security.
- Provide fast repair after a temporary disk failure through Oracle ASM Fast Mirror Resync and automatic repair of block corruptions if a good copy exists in one of the mirrors.
- Provide disaster recovery capability for the file system by enabling replication of Oracle ACFS across the network to a remote site.
- Patch the Oracle ASM instance without impacting the clients that are being serviced using Oracle Flex ASM. A database instance can be directed to access Oracle ASM metadata from another location while the current Oracle ASM instance it is connected to is taken offline for planned maintenance.
- Monitor and manage the speed and status of Oracle ASM Disk Resync and Rebalance operations.
- Bring online multiple disks simultaneously and manage performance better by controlling resync parallelism using the Oracle ASM Resync Power Limit. Recover faster after a cell or disk failure, and the instance doing the resync is failing; this is made possible by using a Disk Resync Checkpoint which enables a resync to resume from where it was interrupted or stopped instead of starting from the beginning.

- Automatically connect database instances to another Oracle ASM instance using Oracle Flex ASM. The local database instance can still access the required metadata and data if an Oracle ASM instance fails due to an unplanned outage.
- Use flex diskgroups to prioritize high availability benefits across multiple databases all using the same diskgroup. Some of the key HA benefits are file extent redundancy, rebalance power limit, and rebalance priority. With flex diskgroups, you can set different values for the above features for different databases, resulting in prioritization across multiple databases within one diskgroup.
- Use flex diskgroups to implement quoto_groups across multiple databases sharing one diskgroup which helps in space management and protection.
- Use flex diskgroups to create point-in-time database clones using the ASM split mirror feature.
- Use preferred reads with stretch clusters to improve performance by affinitizing reads to a site.

> **See Also:**
>
> *Oracle Automatic Storage Management Administrator's Guide*

Fast Recovery Area

The fast recovery area is a unified storage location for all recovery-related files and activities in Oracle Database.

After this feature is enabled, all RMAN backups, archived redo log files, control file autobackups, flashback logs, and data file copies are automatically written to a specified file system or Oracle ASM disk group, and the management of this disk space is handled by RMAN and the database server.

Performing a backup to disk is faster because using the fast recovery area eliminates the bottleneck of writing to tape. More important, if database media recovery is required, then data file backups are readily available. Restoration and recovery time is reduced because you do not need to find a tape and a free tape device to restore the needed data files and archived redo log files.

The fast recovery area provides the following benefits:

- Unified storage location of related recovery files
- Management of the disk space allocated for recovery files, which simplifies database administration tasks
- Fast, reliable, disk-based backup and restoration

> **See Also:**
>
> *Oracle Database Backup and Recovery User's Guide*

Corruption Prevention, Detection, and Repair

Data block corruptions can be very disruptive and challenging to repair. Corruptions can cause serious application and database downtime and data loss when encountered and worse yet it can go undetected for hours, days and even weeks leading to even longer application downtime once detected.Unfortunately, there is not one way to comprehensively prevent, detect, and repair data corruptions within the database because the source and cause of corruptions can be anywhere in memory, hardware, firmware, storage, operating system, software, or user error. Worse yet, third-party solutions that do not understand Oracle data block semantics and how Oracle changes data blocks do not prevent and detect data block corruptions well. Third party remote mirroring technologies can propagate data corruptions to the database replica (standby) leading to a double failure, data loss, and much longer downtime. Third party backup and restore solutions cannot detect corrupted backups or bad sectors until a restore or validate operation is issued, resulting in longer restore times and once again potential data loss.

Oracle MAA has a comprehensive plan to prevent, detect, and repair all forms of data block corruptions including physical block corruptions, logical block corruptions, stray writes, and lost writes. These additional safeguards provide the most comprehensive Oracle data block corruption prevention, detection, and repair solution. Details of this plan are described in the My Oracle Support note "Best Practices for Corruption Detection, Prevention, and Automatic Repair - in a Data Guard Configuration (Doc ID 1302539.1)."

The following outlines block corruption checks for various manual operational checks and runtime and background corruption checks. Database administrators and the operations team can incorporate manual checks such as running Oracle Recovery Manager (RMAN) backups, RMAN "check logical" validations, or running the `ANALYZE VALIDATE STRUCTURE` command on important objects. Manual checks are especially important to validate data that are rarely updated or queried.

Runtime checks are far superior in that they catch corruptions almost immediately or during runtime for actively queried and updated data. Runtime checks can prevent corruptions or automatically fix corruptions resulting in better data protection and higher application availability. A new background check has been introduced in Exadata to automatically scan and scrub disks intelligently with no application overhead and to automatically fix physically corrupted blocks.

Table 3-1 Summary of Block Corruption Checks

Checks	Capabilities	Physical Block Corruption	Logical Block Corruption
Manual checks	Dbverify, Analyze	Physical block checks	Logical intra-block and inter-object consistency checks
Manual checks	RMAN	Physical block checks during backup and restore operations	Intra-block logical checks
Manual checks	ASM Scrub	Physical block checks	Some logical intra-block checks

Table 3-1 (Cont.) Summary of Block Corruption Checks

Checks	Capabilities	Physical Block Corruption	Logical Block Corruption
Runtime checks	Oracle Active Data Guard	1. Continuous physical block checking at standby during transport and apply 2. Strong database isolation eliminates single point database failure 3. Automatic repair of block corruptions, including file block headers in Oracle Database 12*c* Release 2 4. Automatic database failover	1. With `DB_LOST_WRITE_PROTECT` enabled, detection of lost writes (11.2 and higher). With 11.2.0.4 and Data Guard broker, ability to shutdown the primary when lost writes are detected on the primary database. 2. With `DB_BLOCK_CHECKING` enabled on the standby, additional intra-block logical checks
Runtime checks	Database	With `DB_BLOCK_CHECKSUM`, in-memory data block and redo checksum validation	With `DB_BLOCK_CHECKING`, in-memory intra-block check validation Starting in Oracle Database 18c, and with Shadow Lost Write Protection enabled, Oracle tracks system change numbers (SCNs) for tracked data files and enables early lost write detection. When lost writes are detected, an error is returned immediately. See Shadow Lost Write Protection description following this table.
Runtime checks	ASM and ASM software mirroring (inherent in Exadata, Supercluster, and Zero Data Loss Recovery Appliance)	Implicit data corruption detection for reads and writes and automatic repair if good ASM extent block pair is available during writes	.
Runtime checks	DIX + T10 DIF	Checksum validation from operating system to HBA controller to disk (firmware). Validation for reads and writes for certified Linux, HBA and disks.	.
Runtime checks	Hardware and Storage	Limited checks due to lack of Oracle integration. Checksum is most common.	Limited checks due to lack of Oracle integration. Checksum is most common

Table 3-1 (Cont.) Summary of Block Corruption Checks

Checks	Capabilities	Physical Block Corruption	Logical Block Corruption
Runtime checks	Exadata	Comprehensive HARD checks on writes	HARD checks on writes
Background checks	Exadata	Automatic HARD disk scrub and repair. Detects and fixes bad sectors.	.

Shadow Lost Write Protection

New in Oracle Database 18c, shadow lost write protection detects a lost write before it can result in a major data corruption. You can enable shadow lost write protection for a database, a tablespace, or a data file without requiring an Oracle Data Guard standby database. Shadow lost write protection provides fast detection and immediate response to a lost write, thus minimizing the data loss that can occur in a database due to data corruption.

> **See Also:**
>
> *Oracle Database Reference* for more information about the views and initialization parameters
>
> My Oracle Support Note 1302539.1

Data Recovery Advisor

Data Recovery Advisor automatically diagnoses persistent (on-disk) data failures, presents appropriate repair options, and runs repair operations at your request.

You can use Data Recovery Advisor to troubleshoot primary databases, logical standby databases, physical standby databases, and snapshot standby databases.

Data Recovery Advisor includes the following functionality:

- Failure diagnosis

  The first symptoms of database failure are usually error messages, alarms, trace files and dumps, and failed health checks. Assessing these symptoms can be complicated, error-prone, and time-consuming. Data Recovery Advisor automatically diagnoses data failures and informs you about them.

- Failure impact assessment

  After a failure is diagnosed, you must understand its extent and assess its impact on applications before devising a repair strategy. Data Recovery Advisor automatically assesses the impact of a failure and displays it in an easily understood format.

- Repair generation

  Even if a failure was diagnosed correctly, selecting the correct repair strategy can be error-prone and stressful. Moreover, there is often a high penalty for making poor decisions in terms of increased downtime and loss of data. Data Recovery Advisor automatically determines the best repair for a set of failures and presents it to you.

- Repair feasibility checks

  Before presenting repair options, Data Recovery Advisor validates them with respect to the specific environment and availability of media components required to complete the proposed repair, including restoring files directly from the primary or standby database to complete the proposed repair.

- Repair automation

  If you accept the suggested repair option, Data Recovery Advisor automatically performs the repair, verifies that the repair was successful, and closes the appropriate failures.

- Validation of data consistency and database recoverability

  Data Recovery Advisor can validate the consistency of your data, and backups and redo stream, whenever you choose.

- Early detection of corruption

  Through Health Monitor, you can schedule periodic runs of Data Recovery Advisor diagnostic checks to detect data failures before a database process running a transaction discovers the corruption and signals an error. Early warnings can limit the damage caused by corruption.

- Integration of data validation and repair

  Data Recovery Advisor is a single tool for data validation and repair.

> **Note:**
>
> Data Recovery Advisor only supports single-instance databases. Oracle RAC databases are not supported.

> **See Also:**
>
> *Oracle Database Backup and Recovery User's Guide* for information about Data Recovery Advisor supported database configurations.

Oracle Flashback Technology

Oracle Flashback technology is a group of Oracle Database features that let you view past states of database, database objects, transactions or rows or to rewind the database, database objects, transactions or rows to a previous state without using point-in-time media recovery.

With flashback features, you can:

- Perform queries to show data as it looked at a previous point in time
- Perform queries that return metadata that shows a detailed history of changes to the database
- Recover tables or rows to a previous point in time
- Automatically track and archive transactional data changes
- Roll back a transaction and its dependent transactions while the database remains online
- Undrop a table

- Recover a database to a point-in-time without a restore operation

Other than the flashback database feature, most Oracle Flashback features use the Automatic Undo Management (AUM) system to obtain metadata and historical data for transactions. They rely on undo data, which are records of the effects of individual transactions. For example, if a user runs an UPDATE statement to change a salary from 1000 to 1100, then Oracle Database stores the value 1000 in the undo data.

Undo data is persistent and survives a database shutdown. By using flashback features, you can use undo data to query past data or recover from logical damage. Besides using it in flashback features, Oracle Database uses undo data to perform these actions:

- Roll back active transactions
- Recover terminated transactions by using database or process recovery
- Provide read consistency for SQL queries

Oracle Flashback can address and rewind data that is compromised due to various human or operator errors that inadvertently or maliciously change data, cause bad installations and upgrades, and result in logical errors in applications. These problems use features such as flashback transaction, flashback drop, flashback table, and flashback database.

See Also:

Oracle Database Development Guide

Performing Flashback and Database Point-in-Time Recovery, Using Flashback Database and Restore Points, and Performing Block Media Recovery in the *Oracle Database Backup and Recovery User's Guide*

Oracle Database PL/SQL Packages and Types Reference

Oracle Database Backup and Recovery Reference

Oracle Flashback Query

Oracle Flashback Query (Flashback Query) provides the ability to view data as it existed in the past by using the Automatic Undo Management system to obtain metadata and historical data for transactions.

Undo data is persistent and survives a database malfunction or shutdown. The unique features of Flashback Query not only provide the ability to query previous versions of tables, they also provide a powerful mechanism to recover from erroneous operations.

Uses of Flashback Query include:

- Recovering lost data or undoing incorrect, committed changes. For example, rows that were deleted or updated can be immediately repaired even after they were committed.
- Comparing current data with the corresponding data at some time in the past. For example, by using a daily report that shows the changes in data from yesterday, it is possible to compare individual rows of table data, or find intersections or unions of sets of rows.
- Checking the state of transactional data at a particular time, such as verifying the account balance on a certain day.

- Simplifying application design by removing the need to store certain types of temporal data. By using Flashback Query, it is possible to retrieve past data directly from the database.
- Applying packaged applications, such as report generation tools, to past data.
- Providing self-service error correction for an application, enabling users to undo and correct their errors.

Oracle Flashback Version Query

Oracle Flashback Version Query is an extension to SQL that you can use to retrieve the versions of rows in a given table that existed at a specific time interval.

Oracle Flashback Version Query returns a row for each version of the row that existed in the specified time interval. For any given table, a new row version is created each time the `COMMIT` statement is issued.

Oracle Flashback Version Query is a powerful tool that database administrators (database administrators) can use to run analysis to determine the source of problems. Additionally, application developers can use Oracle Flashback Version Query to build customized applications for auditing purposes.

Oracle Flashback Transaction

Oracle Flashback Transaction backs out a transaction and its dependent transactions.

The

```
DBMS_FLASHBACK.TRANSACTION_BACKOUT()
```

procedure rolls back a transaction and its dependent transactions while the database remains online. This recovery operation uses undo data to create and run the compensating transactions that return the affected data to its original state. You can query the

```
DBA_FLASHBACK_TRANSACTION_STATE
```

view to see whether the transaction was backed out using dependency rules or forced out by either:

- Backing out nonconflicting rows
- Applying undo SQL

Oracle Flashback Transaction increases availability during logical recovery by quickly backing out a specific transaction or set of transactions and their dependent transactions. You use one command to back out transactions while the database remains online.

Oracle Flashback Transaction Query

Oracle Flashback Transaction Query provides a mechanism to view all of the changes made to the database at the transaction level.

When used in conjunction with Oracle Flashback Version Query, it offers a fast and efficient means to recover from a human or application error. Oracle Flashback Transaction Query increases the ability to perform online diagnosis of problems in the database by returning the database user that changed the row, and performs analysis and audits on transactions.

Oracle Flashback Table

Oracle Flashback Table recovers a table to a previous point in time.

It provides a fast, online solution for recovering a table or set of tables that were changed by a human or application error. In most cases, Oracle Flashback Table alleviates the need for administrators to perform more complicated point-in-time recovery operations. The data in the original table is not lost when you use Oracle Flashback Table because you can return the table to its original state.

Oracle Flashback Drop

Although there is no easy way to recover dropped tables, indexes, constraints, or triggers, Oracle Flashback Drop provides a safety net when you are dropping objects.

When you drop a table, it is automatically placed into the Recycle Bin. The Recycle Bin is a virtual container where all dropped objects reside. You can continue to query data in a dropped table.

Restore Points

When an Oracle Flashback recovery operation is performed on the database, you must determine the point in time—identified by the system change number (SCN) or time stamp—to which you can later flash back the data.

Oracle Flashback restore points are labels that you can define to substitute for the SCN or transaction time used in Flashback Database, Flashback Table, and Oracle Recovery Manager (RMAN) operations. Furthermore, a database can be flashed back through a previous database recovery and opened with an

```
OPEN RESETLOGS
```

command by using guaranteed restore points. Guaranteed restore points allow major database changes—such as database batch jobs, upgrades, or patches—to be quickly undone by ensuring that the undo required to rewind the database is retained.

Using the restore points feature provides the following benefits:

- The ability to quickly restore to a consistent state, to a time before a planned operation that has gone awry (for example, a failed batch job, an Oracle software upgrade, or an application upgrade)
- The ability to resynchronize a snapshot standby database with the primary database
- A quick mechanism to restore a test or cloned database to its original state

Oracle Flashback Database

Oracle Flashback Database is the equivalent of a fast rewind button, quickly returning a database to a previous point in time without requiring a time consuming restore and roll forward using a backup and archived logs.

The larger the size of the database, the greater the advantage of using Oracle Flashback Database for fast point in time recovery.

Enabling Oracle Flashback Database provides the following benefits:

- Fast point in time recovery to repair logical corruptions, such as those caused by administrative error.
- Useful for iterative testing when used with Oracle restore points. A restore point can be set, database changes implemented, and test workload run to assess impact. Oracle Flashback Database can then be used to discard the changes and return the database to the original starting point, different modifications can be made, and the same test workload run a second time to have a true basis for comparing the impact of the different configuration changes.
- Data Guard uses Oracle Flashback Database to quickly reinstantiate a failed primary database as a new standby (after a failover has occurred), without requiring the failed primary to be restored from a backup.
- Flashback database operates at the CDB level or the PDB level.

Flashback Pluggable Database

You can rewind a PDB to a previous SCN. The `FLASHBACK PLUGGABLE DATABASE` command, which is available through SQL or Recovery Manager, is analogous to `FLASHBACK DATABASE` in a non-CDB.

Flashback PDB protects an individual PDB against data corruption, widespread user errors, and redo corruption. The operation does not rewind data in other PDBs in the CDB.

You can use

```
CREATE RESTORE POINT ... FOR PLUGGABLE
        DATABASE
```

to create a PDB restore point, which is only usable within a specified PDB. As with CDB restore points, PDB restore points can be normal or guaranteed. A guaranteed restore point never ages out of the control file and must be explicitly dropped. If you connect to the root, and if you do not specify the

```
FOR PLUGGABLE
        DATABASE
```

clause, then you create a CDB restore point, which is usable by all PDBs.

A special type of PDB restore point is a clean restore point, which you can only create when a PDB is closed. For PDBs with shared undo, rewinding the PDB to a clean restore point is faster than other options because it does not require restoring backups or creating a temporary database instance.

Block Media Recovery Using Flashback Logs or Physical Standby Database

After attempting to automatically repair corrupted blocks, block media recovery can optionally retrieve a more recent copy of a data block from the flashback logs to reduce recovery time.

Automatic block repair allows corrupt blocks on the primary database to be automatically repaired as soon as they are detected, by using good blocks from a physical standby database.

Furthermore, a corrupted block encountered during instance recovery does not result in instance recovery failure. The block is automatically marked as corrupt and added to the RMAN corruption list in the

`V$DATABASE_BLOCK_CORRUPTION`

table. You can subsequently issue the RMAN

`RECOVER BLOCK`

command to fix the associated block. In addition, the RMAN

`RECOVER BLOCK`

command restores blocks from a physical standby database, if it is available.

Flashback Data Archive

The Flashback Data Archive is stored in a tablespace and contains transactional changes to every record in a table for the duration of the record's lifetime.

The archived data can be retained for a much longer duration than the retention period offered by an undo tablespace, and used to retrieve very old data for analysis and repair.

Oracle Data Pump and Data Transport

Oracle Data Pump technology enables very high-speed movement of data and metadata from one database to another. Data Pump is used to perform the following planned maintenance activities:

- Database migration to a different platform
- Database migration to pluggable databases
- Database upgrade

The Data Pump features that enable the planned maintenance activities listed above are the following:

- Full transportable export/import to move an entire database to a different database instance
- Transportable tablespaces to move a set of tablespaces between databases

> **See Also:**
>
> Transporting Data

Oracle Replication Technologies for Non-Database Files

Oracle Advanced Cluster File System (Oracle ACFS), Oracle Database File System, and Oracle Solaris ZFS Storage Appliance Replication are the Oracle replication technologies for non-database files.

Table 3-2 Oracle Replication Technologies for Non-Database Files

Technology	Recommended Usage	Comments
Oracle Advanced Cluster File System	Recommended to provide a single-node and cluster-wide file system solution integrated with Oracle ASM, Oracle Clusterware, and Oracle Enterprise Manager technologies. Provides a loosely coupled full stack replication solution when combined with Data Guard or Oracle GoldenGate.	Oracle ACFS establishes and maintains communication with the Oracle ASM instance to participate in Oracle ASM state transitions including Oracle ASM instance and disk group status updates and disk group rebalancing. Supports many database and application files, including executables, database trace files, database alert logs, application reports, BFILEs, and configuration files. Other supported files are video, audio, text, images, engineering drawings, and other general-purpose application file data. Can provide near-time consistency between database changes and file system changes when point-in-time recovery happens Can be exported and accessed by remote clients using standard NAS File Access Protocols such as NFS and CIFS.
Oracle Database File System	Recommended for providing stronger synchronization between database and non-database systems.	Can be integrated with the database to maintain complete consistency between the database changes and the file system changes All data stored in the database and can be used with Oracle Active Data Guard to provide both disaster recovery and read-only access Can take advantage all of the Oracle database features
Oracle Solaris ZFS Storage Appliance Replication	Recommended for disaster recovery protection for non-database files, and specifically for Oracle Fusion Middleware critical files stored outside of the database.	Replicates all non-database objects, including Oracle Fusion Middleware binaries configuration Can provide near time consistency between database changes and file system changes when point-in-time recovery happens

Oracle Advanced Cluster File System

Oracle Advanced Cluster File System (ACFS) is a multiplatform, scalable file system, and storage management technology that extends Oracle Automatic Storage Management (Oracle ASM) functionality to support customer files maintained outside of Oracle Database.

Oracle ACFS supports many database and application files, including executables, database trace files, database alert logs, application reports, BFILEs, and configuration files. Other

supported files are video, audio, text, images, engineering drawings, and other general-purpose application file data.

Oracle ACFS takes advantage of the following Oracle ASM functionality:

- Oracle ACFS dynamic file system resizing
- Maximized performance through direct access to Oracle ASM disk group storage
- Balanced distribution of Oracle ACFS across Oracle ASM disk group storage for increased I/O parallelism
- Data reliability through Oracle ASM mirroring protection mechanisms

Oracle ACFS Replication, similar to Data Guard for the database, enables replication of Oracle ACFS file systems across the network to a remote site, providing disaster recovery capability for the file system. Oracle ACFS replication captures file system changes written to disk for a primary file system and records the changes in files called replication logs. These logs are transported to the site hosting the associated standby file system where background processes read the logs and apply the changes recorded in the logs to the standby file system. After the changes recorded in a replication log are successfully applied to the standby file system, the replication log is deleted from the sites hosting the primary and standby file systems.

An additional feature of Oracle ACFS is that it offers snapshot-based replication for generic and application files, providing an HA solution for disaster recovery and Test/Development environments. Oracle Databases stored in ACFS can leverage Oracle Multiltenant and ACFS snapshot technologies to create quick and efficient snapshot clones of pluggable databases.

Oracle Data Guard and Oracle ACFS can be combined to provide a full stack high availability solution with Data Guard protecting the database with a standby database and Oracle ACFS replicating the file system changes to the standby host. For planned outages the file system and the database remain consistent to a point in time with zero data loss.

> **See Also:**
>
> Oracle ACFS ASM Cluster File System: What is it and How to use it
>
> http://www.oracle.com/goto/maa for Oracle MAA white paper “Full Stack Role Transition - Oracle ACFS and Oracle Data Guard”

Oracle Database File System

Oracle Database File System (DBFS) takes advantage of the features of the database to store files, and the strengths of the database in efficiently managing relational data, to implement a standard file system interface for files stored in the database.

With this interface, storing files in the database is no longer limited to programs specifically written to use BLOB and CLOB programmatic interfaces. Files in the database can now be transparently accessed using any operating system (OS) program that acts on files. For example, extract, transform, and load (ETL) tools can transparently store staging files in the database.

Oracle DBFS provides the following benefits:

- Full stack integration recovery and failover: By storing file system files in a database structure, it is possible to easily perform point-in-time recovery of both database objects and file system data.

- Disaster Recovery System Return on Investment (ROI): All changes to files contained in DBFS are also logged through the Oracle database redo log stream and thus can be passed to a Data Guard physical standby database. Using Oracle Active Data Guard technology, the DBFS file system can be mounted read-only using the physical standby database as the source. Changes made on the primary are propagated to the standby database and are visible once applied to the standby.
- File system backups: Because DBFS is stored in the database as database objects, standard RMAN backup and recovery functionality can be applied to file system data. Any backup, restore, or recovery operation that can be performed on a database or object within a database can also be performed against the DBFS file system.

> **See Also:**
>
> Database File System (DBFS)

Oracle Solaris ZFS Storage Appliance Replication

The Oracle Solaris ZFS Storage Appliance series supports snapshot-based replication of projects and shares from a source appliance to any number of target appliances manually, on a schedule, or continuously.

The Oracle Solaris ZFS Storage Appliance series supports the following use cases:

- Disaster recovery: Replication can be used to mirror an appliance for disaster recovery. In the event of a disaster that impacts the service of the primary appliance (or even an entire data center), administrators activate the service at the disaster recovery site, which takes over using the most recently replicated data. When the primary site is restored, data changed while the disaster recovery site was in service can be migrated back to the primary site, and normal service is restored. Such scenarios are fully testable before a disaster occurs.
- Data distribution: Replication can be used to distribute data (such as virtual machine images or media) to remote systems across the world in situations where clients of the target appliance would not ordinarily be able to reach the source appliance directly, or such a setup would have prohibitively high latency. One example uses this scheme for local caching to improve latency of read-only data (such as documents).
- Disk-to-disk backup: Replication can be used as a backup solution for environments in which tape backups are not feasible. Tape backup might not be feasible, for example, because the available bandwidth is insufficient or because the latency for recovery is too high.
- Data migration: Replication can be used to migrate data and configuration between Oracle Solaris ZFS Storage appliances when upgrading hardware or rebalancing storage. Shadow migration can also be used for this purpose.

The architecture of Oracle Solaris ZFS Storage Appliance also makes it an ideal platform to complement Data Guard for disaster recovery of Oracle Fusion Middleware. Oracle Fusion Middleware has a number of critical files that are stored outside of the database. These binaries, configuration data, metadata, logs and so on also require data protection to ensure availability of the Oracle Fusion Middleware. For these, the built-in replication feature of the ZFS Storage Appliance is used to move this data to a remote disaster recovery site.

Benefits of the Oracle Solaris ZFS Storage Appliance when used with Oracle Fusion Middleware include:

- Leverages remote replication for Oracle Fusion Middleware
- Provides ability to quickly create clones and snapshots of databases to increase ROI of DR sites

> **See Also:**
>
> Oracle ZFS Storage Appliance Software

Oracle Multitenant

Oracle Multitenant is the optimal database consolidation method. The multitenant architecture combines the best attributes of each of the previous consolidation methods without their accompanying tradeoffs.

Oracle Multitenant helps reduce IT costs by simplifying consolidation, provisioning, upgrades and more. This new architecture allows a container database (CDB) to hold many pluggable databases (PDBs). To applications, these PDBs appear as a standalone database, and no changes are required to the application in order to access the PDB. By consolidating multiple databases as PDBs into a single CDB, you are provided with the ability to manage "many as one". The flexibility remains to operate on PDBs in isolation should your business require it.

Oracle Multitenant is fully compliant with and takes direct advantage of high availability features such as Oracle Real Application Clusters, Oracle Data Guard, and Oracle GoldenGate, just like any non-container database (non-CDB), meaning it can be used in any of the Oracle MAA reference architectures. Grouping multiple PDBs with the same high availability requirements into the same CDB ensures that all of those PDBs and their applications are managed and protected with the same technologies and configurations.

Benefits of Using Oracle Multitenant

- High consolidation density - Many PDBs can be stored in a single CDB. These PDBs share background processes and memory structures letting you run more PDBs than you would non-CDBs, because the overhead for each non-CDB is removed or reduced. You can store up to 4095 PDBs in a CDB. Each PDB can also have a different characterset from other PDBs within the same CDB, as long as the CDB root character set is a superset of all of the PDBs' character sets. Logical standby databases also support this mix of character sets to allow rolling upgrades with a transient logical standby database.
- Online provisioning operations, including clones, refreshable clones, and PDB relocation - A PDB can be unplugged from one CDB and plugged into another. A PDB can also be cloned either into the same CDB or into a different CDB. Cloning can be used to create a "gold image" or seed database for DBaaS or SaaS environments. This PDB can then be rapidly cloned to easily set up database environments for new customers.
  - Near Zero Downtime PDB Relocation – This feature significantly reduces the downtime of relocating a PDB from one CDB to another by using clone functionality. The source PDB remains open and functional while the relocation takes place. The application outage is reduced to a very short window while the source PDB is brought to a consistent state, and the destination PDB is synchronized and brought online. This functionality also takes advantage of another new feature, Listener Redirects, which allows you to keep the same connect descriptor for applications and connect to the destination PDB even after it has been relocated.
  - Online provisioning and cloning – Clones of PDBs can be created without requiring the source PDB to be placed in read only-mode. The source PDB can be left in read-write mode and accessible to applications for the duration of the clone operation.

  - Refreshable Clone PDB – Clones of PDBs can be created in such a way as to be refreshed with changes with changes made to the source PDB applied either automatically at set intervals or manually. For a clone to be refreshable it must remain in read-only mode. The clone can be converted into an ordinary PDB by opening it read-write. Refreshable clones are well suited to be used as test masters for Exadata storage snapshots.

- New patching and upgrade options -When you upgrade or patch a CBD, all of the PDBs in that container are also upgraded or patched. If you need isolation, you can unplug a PDB and plug it into a CDB at a later version.
- Database backup and recovery - By consolidating multiple databases as PDBs, operations such as backup and disaster recovery are performed at the container level. Oracle Multitenant also provides the flexibility to backup and restore individual PDBs with no impact to other running PDBs in the same CDB.
- Operation with Oracle Data Guard - Data Guard configurations are maintained at the CDB level. When a Data Guard role transition (either failover or switchover) is performed, all PDBs are transitioned to the new primary database. There is no need to create or manage multiple Data Guard configurations for each PDB as would be required for single databases. Existing tools such as Data Guard Standby First Patching and Data Guard Transient Logical Rolling Upgrade can still be used to reduce downtime and are performed at the container level, so all PDBs will be maintained in a single operation.
  - PDB Migration with Data Guard Broker – The Data Guard broker has been enhanced to provide automation for migrating PDBs from one CDB, either the primary database or the standby database, to another CDB. This can be used for straight migration of a PDB from one CDB to another running at either at the same version or a CDB running at a higher version to start the upgrade process. This automation can also be used to affect a single PDB failover by using the PDBs files at a standby database to plug into a different CDB at the same version.
  - Subset Standby - A subset standby enables users of Oracle Multitenant to designate a subset of the PDBs in a CDB for replication to a standby database. This provides a finer granularity of designating which standby databases will contain which PDBs.
- Operation with Oracle GoldenGate - All of functionality provided by Oracle GoldenGate also exists for Oracle Multitenant. GoldenGate also provides the flexibility to operate at the PDB level, allowing replication to occur for a subset of the PDBs in a CDB. GoldenGate can be used for minimal to zero downtime upgrades either at the CDB level or at an individual PDB level.
- Resource management - Just as Oracle Resource Manager can control resource utilization between single databases, it can also control resource utilization of individual PDBs in a container. This can ensure that a single PDB does not access more than its assigned share of system resources. You can specify guaranteed minimums and maximums for SGA, buffer cache, shared pool, and PGA memory at the PDB limit.
- Operation with Oracle Flashback Database - If fast point-in-time recovery is required, the initial release of Oracle Multitenant enables using Flashback Database at the CDB level. Oracle Multitenant enables Flashback Database to be used on an individual PDB without impacting the availability of other PDBs. Flashback Database can performed at the CDB level which will flashback all of the PDBs in the container. Individual PDBs can be flashed back using the Flashback Pluggable Database feature. When flashing back an individual PDB all other PDBs remain unaffected.
- Data Guard Broker PDB Migration or Failover - In multitenant broker configurations, you may need to move a Production PDB from one container database to another container database that resides on the same system. You may also need to failover a PDB from a Data Guard Standby database to a new production container database when the production PDB has failed but the container database and all other PDBs function

normally. Using the new Data Guard Broker command, `MIGRATE PLUGGABLE DATABASE`, you can easily move a single PDB from one container database to another, or failover a single PDB from a Data Guard standby to a new production container database. (new in Oracle Database 12c Release 2)

See Also:

- *Oracle Multitenant Administrator's Guide*
- Oracle MAA technical brief "Best Practices for Database Consolidation" at https://www.oracle.com/database/technologies/high-availability/oracle-database-maa-best-practices.html

Oracle Sharding

Oracle Sharding is a scalability and availability feature for applications explicitly designed to run on a sharded database.

Oracle sharding enables distribution and replication of data across a pool of Oracle databases that share no hardware or software. The pool of databases is presented to the application as a single logical database. Applications elastically scale (data, transactions, and users) to any level, on any platform, simply by adding additional databases (shards) to the pool. Scaling up to 1000 shards is supported.

Oracle Sharding provides superior run-time performance and simpler life-cycle management compared to home-grown deployments that use a similar approach to scalability. It also provides the advantages of an enterprise DBMS, including relational schema, SQL, and other programmatic interfaces, support for complex data types, online schema changes, multi-core scalability, advanced security, compression, high-availability, ACID properties, consistent reads, developer agility with JSON, and much more.

See Also:

Using Oracle Sharding

Oracle Restart

Oracle Restart enhances the availability of a single-instance (nonclustered) Oracle database and its components.

Oracle Restart is used in single-instance environments only. For Oracle Real Application Clusters (Oracle RAC) environments, the functionality to automatically restart components is provided by Oracle Clusterware.

If you install Oracle Restart, it automatically restarts the database, the listener, and other Oracle components after a hardware or software failure or whenever the database's host computer restarts. It also ensures that the Oracle components are restarted in the proper order, in accordance with component dependencies.

Oracle Restart periodically monitors the health of components—such as SQL*Plus, the Listener Control utility (LSNRCTL), ASMCMD, and Oracle Data Guard—that are integrated

with Oracle Restart. If the health check fails for a component, Oracle Restart shuts down and restarts the component.

Oracle Restart runs out of the Oracle Grid Infrastructure home, which you install separately from Oracle Database homes.

Integrated client failover applications depend on role based services and Fast Application Notification events, managed by Oracle clusterware, to alert the application to failures. Single instance databases must have Oracle Restart to achieve integrated client failover.

> **See Also:**
>
> *Oracle Database Administrator's Guide* for information about installing and configuring the Oracle Restart feature

Online Reorganization and Redefinition

One way to enhance availability and manageability is to allow user access to the database during a data reorganization operation.

The Online Reorganization and Redefinition feature in Oracle Database offers administrators significant flexibility to modify the physical attributes of a table and transform both data and table structure while allowing user access to the database. This capability improves data availability, query performance, response time, and disk space usage. All of these are important in a mission-critical environment and make the application upgrade process easier, safer, and faster.

Use Oracle Database online maintenance features to significantly reduce (or eliminate) the application downtime required to make changes to an application's database objects

> **See Also:**
>
> Redefining Tables Online in *Oracle Database Administrator's Guide*

Zero Data Loss Recovery Appliance

The cloud-scale Zero Data Loss Recovery Appliance, commonly known as Recovery Appliance, is an engineered system designed to dramatically reduce data loss and backup overhead for all Oracle databases in the enterprise.

Integrated with Recovery Manager (RMAN), the Recovery Appliance enables a centralized, incremental-forever backup strategy for large numbers of databases, using cloud-scale, fault-tolerant hardware and storage. The Recovery Appliance continuously validates backups for recoverability.

Recovery Appliance is the MAA-preferred backup and recovery appliance because:

- Elimination of data loss when restoring from Recovery Appliance
- Minimal backup overhead
- Improved end-to-end data protection visibility
- Cloud-scale protection

- Integrates very well with all MAA reference architectures including Oracle Sharding tier

> **See Also:**
>
> Zero Data Loss Recovery Appliance Documentation

Fleet Patching and Provisioning

Fleet Patching and Provisioning maintains a space-efficient repository of software, more precisely "gold images," which are standardized software homes that can be provisioned to any number of target machines.

Any number of homes can be provisioned from a given gold image, and Fleet Patching and Provisioning maintains lineage information so that the provenance of deployed software is always known. Gold images can be organized into series, allowing you to create groupings that track the evolution of a release, with different series for different tailored solutions such as Oracle Database patch bundles for specific applications. A notification system informs interested parties when a new image is available in a given series. Fleet Patching and Provisioning is a feature of Oracle Grid Infrastructure. The components that form the Fleet Patching and Provisioning Server are managed automatically by Oracle Grid Infrastructure.

Fleet Patching and Provisioning can provision databases, clusterware, middleware, and custom software. Fleet Patching and Provisioning offers additional features for creating, configuring, patching and upgrading Oracle Grid Infrastructure and Oracle Database deployments. These capabilities simplify maintenance, reducing its risk and impact, and provide a roll-back option if changes need to be backed out. Additional capabilities include provisioning clusters and databases onto base machines, and simple capacity on demand by growing and shrinking clusters and Oracle RAC databases. All of these operations are performed with single commands which replace the numerous manual steps otherwise required. All commands and their outcomes are recorded in an audit log. All workflows allow customization to support the unique requirements of any environment.

The key benefits of Fleet Patching and Provisioning are:

- Enables and enforces standardization
- Simplifies provisioning, patching and upgrading
- Minimizes the impact and risk of maintenance
- Increases automation and reduces touch points
- Supports large scale deployments

See Also:

Fleet Patching and Provisioning and Maintenance in *Oracle Clusterware Administration and Deployment Guide*

Oracle Fleet Patching and Provisioning (FPP) Introduction and Technical Overview

Edition-Based Redefinition

Planned application changes may include changes to data, schemas, and programs. The primary objective of these changes is to improve performance, manageability, and functionality. An example is an application upgrade.

Edition-based redefinition (EBR) lets you upgrade the database component of an application while it is in use, thereby minimizing or eliminating downtime. To upgrade an application while it is in use, you must copy the database objects that comprise the database component of the application and redefine the copied objects in isolation. Your changes do not affect users of the application; they can continue to run the unchanged application. When you are sure that your changes are correct, you make the upgraded application available to all users.

> **See Also:**
>
> Using Edition-Based Redefinition in *Oracle Database Development Guide*

4

Oracle Database High Availability Solutions for Unplanned Downtime

Oracle Database offers an integrated suite of high availability solutions that increase availability.

These solutions also **eliminate or minimize** both planned and unplanned downtime, and help enterprises maintain business continuity 24 hours a day, 7 days a week. However, Oracle's high availability solutions not only go beyond reducing downtime, but also help to improve overall performance, scalability, and manageability.

Outage Types and Oracle High Availability Solutions for Unplanned Downtime

Various Oracle MAA high availability solutions for unplanned downtime are described here in an easy to navigate matrix.

The following table shows how the features discussed in the referenced (hyperlinked) sections can be used to address various causes of unplanned downtime. Where several Oracle solutions are listed, the MAA recommended solution is indicated in the Oracle MAA Solution column.

Table 4-1 Outage Types and Oracle High Availability Solutions for Unplanned Downtime

Outage Scope	Oracle MAA Solution	Benefits
Site failures	Oracle Data Guard and Enabling Continuous Service for Applications (MAA recommended)	• Integrated client and application failover • Fastest and simplest database replication • Supports all data types • Zero data loss by eliminating propagation delay • Oracle Active Data Guard – Supports read-only services and DML on global temporary tables and sequences to off-load more work from the primary – Allows small updates to be redirected to the primary enabling read-mostly reports to be offloaded to standby • Database In-Memory support
	Oracle GoldenGate	• Flexible logical replication solution (target is open read/write) • Active-active high availability (with conflict resolution) • Heterogeneous platform and heterogeneous database support • Potential zero downtime with custom application failover

Table 4-1 (Cont.) Outage Types and Oracle High Availability Solutions for Unplanned Downtime

Outage Scope	Oracle MAA Solution	Benefits
	Recovery Manager, Zero Data Loss Recovery Appliance and Oracle Secure Backup	• Fully managed database recovery and integration with Oracle Secure Backup • Recovery Appliance – provides end-to-end data protection for backups – reduces data loss for database restores – Non-real-time recovery
Instance or computer failures	Oracle Real Application Clusters and Oracle Clusterware and Enabling Continuous Service for Applications (MAA recommended)	• Integrated client and application failover • Automatic recovery of failed nodes and instances • Lowest application brownout with Oracle Real Application Clusters
	Oracle RAC One Node and Enabling Continuous Service for Applications	• Integrated client and application failover • Online database relocation migrates connections and instances to another node • Better database availability than traditional cold failover solutions
	Oracle Data Guard and Enabling Continuous Service for Applications	• Integrated client and application failover • Fastest and simplest database replication • Supports all data types • Zero data loss by eliminating propagation delay • Oracle Active Data Guard – Supports read-only services and DML on global temporary tables and sequences to off-load more work from the primary – Allows small updates to be redirected to the primary enabling read-mostly reports to be offloaded to standby • Database In-Memory support
	Oracle GoldenGate	• Flexible logical replication solution (target is open read/write) • Active-Active high availability (with conflict resolution) • Heterogeneous platform and heterogeneous database support • Potential zero downtime with custom application failover
Storage failures	Oracle Automatic Storage Management (MAA recommended)	Mirroring and online automatic rebalancing places redundant copies of the data in separate failure groups.
	Oracle Data Guard (MAA recommended)	• Integrated client and application failover • Fastest and simplest database replication • Supports all data types • Zero data loss by eliminating propagation delay • Oracle Active Data Guard supports read-only services and DML on global temporary tables and sequences to off-load more work from the primary • Database In-Memory support

Table 4-1 (Cont.) Outage Types and Oracle High Availability Solutions for Unplanned Downtime

Outage Scope	Oracle MAA Solution	Benefits
	Recovery Manager with Fast Recovery Area, and Zero Data Loss Recovery Appliance (MAA recommended)	Fully managed database recovery and managed disk and tape backups
	Oracle GoldenGate	• Flexible logical replication solution (target is open read/write) • Active-active high availability (with conflict resolution) • Heterogeneous platform and heterogeneous database support • Potential zero downtime with custom application failover
Data corruption	Corruption Prevention, Detection, and Repair (MAA recommended) Database initialization settings such as `DB_BLOCK_CHECKING`, `DB_BLOCK_CHECKSUM`, and `DB_LOST_WRITE_PROTECT`	Different levels of data and redo block corruption prevention and detection at the database level

Table 4-1 (Cont.) Outage Types and Oracle High Availability Solutions for Unplanned Downtime

Outage Scope	Oracle MAA Solution	Benefits
Data corruption	Oracle Data Guard (MAA recommended) Oracle Active Data Guard Automatic Block Repair `DB_LOST_WRITE_PROTECT` initialization parameter	• In a **Data Guard** configuration with an **Oracle Active Data Guard** standby – Physical block corruptions detected by Oracle at a primary database are automatically repaired using a good copy of the block retrieved from the standby, and vice versa – The repair is transparent to the user and application, and data corruptions can definitely be isolated • With **MAA recommended initialization settings,** Oracle Active Data Guard and Oracle Exadata Database Machine, achieve most comprehensive full stack corruption protection. • With **`DB_LOST_WRITE_PROTECT`** enabled – A lost write that occurred on the primary database is detected either by the physical standby database or during media recovery of the primary database, recovery is stopped to preserve the consistency of the database – Failing over to the standby database using Data Guard will result in some data loss – Data Guard Broker's `PrimaryLostWrite` property supports SHUTDOWN and CONTINUE, plus FAILOVER and FORCEFAILOVER options, when lost writes are detected on the primary database. See *Oracle Data Guard Broker* – `DB_LOST_WRITE_PROTECT` initialization parameter provides lost write detection • **Shadow lost write protection** detects a lost write before it can result in major data corruption. You can enable shadow lost write protection for a database, a tablespace, or a data file without requiring an Oracle Data Guard standby database. Note the impact on your workload may vary.

Table 4-1 (Cont.) Outage Types and Oracle High Availability Solutions for Unplanned Downtime

Outage Scope	Oracle MAA Solution	Benefits
	Dbverify, Analyze, Data Recovery Advisor and Recovery Manager, Zero Data Loss Recovery Appliance, and ASM Scrub with Fast Recovery Area (MAA recommended)	These tools allow the administrator to run manual checks to help detect and potentially repair from various data corruptions. • **Dbverify** and **Analyze** conduct physical block and logical intra-block checks. Analyze can conduct inter-object consistency checks. • **Data Recovery Advisor** automatically detects data corruptions and recommends the best recovery plan. • **RMAN** operations can – Conduct both physical and inter-block logical checks – Run online block-media recovery using flashback logs, backups, or the standby database to help recover from physical block corruptions • **Recovery Appliance** – Does periodic backup validation that helps ensure that your backups are valid – Allows you to input your recovery window requirements, and alerts you when those SLAs cannot be met with your existing backups managed by Recovery Appliance • **ASM Scrub** detects and attempts to repair physical and logical data corruptions with the ASM pair in normal and high redundancy disks groups.
Data corruption	Oracle Exadata Database Machine and Oracle Automatic Storage Management (MAA recommended) DIX + T10 DIF Extensions (MAA recommended where applicable)	• If Oracle ASM detects a corruption and has a good mirror, ASM returns the good block and repairs the corruption during a subsequent write I/O. • Exadata provides implicit HARD enabled checks to prevent data corruptions caused by bad or misdirected storage I/O. • Exadata provides automatic HARD disk scrub and repair. Detects and fixes bad sectors. • DIX +T10 DIF Extensions provides end to end data integrity for reads and writes through a checksum validation from a vendor's host adapter to the storage device
	Oracle GoldenGate	• Flexible logical replication solution (target is open read/write). Logical replica can be used as a failover target if partner replica is corrupted. • Active-active high availability (with conflict resolution) • Heterogeneous platform and heterogeneous database support
Human errors	Oracle security features (MAA recommended)	Restrict access to prevent human errors
	Oracle Flashback Technology (MAA recommended)	• Fine-grained error investigation of incorrect results • Fine-grained and database-wide or pluggable database rewind and recovery capabilities

Table 4-1 (Cont.) Outage Types and Oracle High Availability Solutions for Unplanned Downtime

Outage Scope	Oracle MAA Solution	Benefits
Delays or slow downs	Oracle Database and Oracle Enterprise Manager Oracle Data Guard (MAA recommended) and Enabling Continuous Service for Applications	• **Oracle Database** automatically monitors for instance and database delays or cluster slow downs and attempts to remove blocking processes or instances to prevent prolonged delays or unnecessary node evictions. • **Oracle Enterprise Manager** or a customized application heartbeat can be configured to detect application or response time slowdown and react to these SLA breaches. For example, you can configure the Enterprise Manager Beacon to monitor and detect application response times. Then, after a certain threshold expires, Enterprise Manager can call the Data Guard `DBMS_DG.INITIATE_FS_FAILOVER` PL/SQL procedure to initiate a failover. See the section about "Managing Fast-Start Failover" in *Oracle Data Guard Broker*. • Database In-Memory support
File system data	Oracle Replication Technologies for Non-Database Files	Enables full stack failover that includes non-database files

Managing Unplanned Outages for MAA Reference Architectures and Multitenant Architectures

High availability solutions in each of the MAA service-level tiers for the MAA reference architectures and multitenant architectures are described in an easy to navigate matrix.

If you are managing many databases in DBaaS, we recommend using the MAA tiers and Oracle Multitenant as described in Oracle MAA Reference Architectures.

The following table identifies various unplanned outages that can impact a database in a multitenant architecture. It also identifies the Oracle high availability solution to address that outage that is available in each of the MAA reference architectures.

Table 4-2 Unplanned Outage Matrix for MAA Reference Architectures and Multitenant Architectures

Event	Solutions by MAA Architecture	Recovery Window (RTO)	Data Loss (RPO)
Instance Failure	BRONZE: Oracle Restart	Minutes if instance can restart	Zero

Table 4-2 (Cont.) Unplanned Outage Matrix for MAA Reference Architectures and Multitenant Architectures

Event	Solutions by MAA Architecture	Recovery Window (RTO)	Data Loss (RPO)
	SILVER: Oracle RAC (see Oracle Real Application Clusters and Oracle Clusterware) or Oracle RAC One Node, and Enabling Continuous Service for Applications	Seconds with Oracle RAC, minutes with Oracle RAC One Node	Zero
	GOLD: Oracle RAC (see Oracle Real Application Clusters and Oracle Clusterware and Enabling Continuous Service for Applications	Seconds	Zero
	PLATINUM: Oracle RAC (see Oracle Real Application Clusters and Oracle Clusterware) and Enabling Continuous Service for Applications	Zero Application Outage	Zero
Permanent Node Failure (but storage available)	BRONZE: Restore and recover	Hours to Day	Zero
	SILVER: Oracle RAC (see Oracle Real Application Clusters and Oracle Clusterware) and Enabling Continuous Service for Applications	Seconds	Zero
	SILVER: Oracle RAC One Node and Enabling Continuous Service for Applications	Minutes	Zero
	GOLD: Oracle RAC (see Oracle Real Application Clusters and Oracle Clusterware) and Enabling Continuous Service for Applications	Seconds	Zero

Table 4-2 (Cont.) Unplanned Outage Matrix for MAA Reference Architectures and Multitenant Architectures

Event	Solutions by MAA Architecture	Recovery Window (RTO)	Data Loss (RPO)
	PLATINUM: Oracle RAC (see Oracle Real Application Clusters and Oracle Clusterware) and Enabling Continuous Service for Applications	Seconds	Zero
Storage Failure	ALL: Oracle Automatic Storage Management	Zero downtime	Zero
Data corruptions	BRONZE/SILVER: Basic protection Some corruptions require recover restore and recovery of pluggable database (PDB), entire multitenant container database (CDB) or non-container database (non-CDB)	Hour to Days	• Since last backup if unrecoverable • Zero or Near Zero with Recovery Appliance
	GOLD: Comprehensive corruption protection and Auto Block Repair with Oracle Active Data Guard	• Zero with auto block repair • Seconds to minutes if corruption due to lost writes and using Data Guard Fast Start failover.	Zero unless corruption due to lost writes
	PLATINUM: Comprehensive corruption protection and Auto Block Repair with Oracle Active Data Guard Oracle GoldenGate replica with custom application failover	• Zero with auto block repair • Zero with Oracle GoldenGate replica	Zero when using Active Data Guard Fast-Start Failover and Oracle GoldenGate
Human error	ALL: Logical failures resolved by flashback drop, flashback table, flashback transaction, flashback query flashback pluggable database, and undo.	Dependent on detection time but isolated to PDB and applications using those objects.	Dependent on logical failure

Table 4-2 (Cont.) Unplanned Outage Matrix for MAA Reference Architectures and Multitenant Architectures

Event	Solutions by MAA Architecture	Recovery Window (RTO)	Data Loss (RPO)
	All: Comprehensive logical failures impacting an entire database and PDB that requires RMAN point in time recovery (PDB) or flashback pluggable database	Dependent on detection time	Dependent on logical failure
Database unusable, system, site or storage failures, wide spread corruptions or disasters	BRONZE/SILVER: Restore and recover	Hours to Days	• Since last database and archive backup • Zero or near zero with Recovery Appliance
	GOLD: Active Data Guard Fast-Start Failover and Enabling Continuous Service for Applications	Seconds	Zero to Near Zero
	PLATINUM: Oracle GoldenGate replica with custom application failover	Zero	Zero when using Active Data Guard Fast-Start Failover and Oracle GoldenGate
Performance Degradation	ALL: Oracle Enterprise Manager for monitoring and detection, Database Resource Management for Resource Limits and ongoing Performance Tuning	No downtime but degraded service	Zero

5

Oracle Database High Availability Solutions for Planned Downtime

Planned downtime can be just as disruptive to operations as unplanned downtime. This is especially true for global enterprises that must support users in multiple time zones, or for those that must provide Internet access to customers 24 hours a day, 7 days a week.

See the following topics to learn about keeping your database highly available during planned downtime.

Oracle High Availability Solutions for Planned Maintenance

Oracle provides high availability solutions for all planned maintenance.

The following table describes the various Oracle high availability solutions and their projected downtime for various maintenance activities.

Table 5-1 Oracle High Availability Solutions for Planned Maintenance

Maintenance Event	High Availability Solutions with Target Outage Time
Dynamic and Online Resource Provisioning, or Online reorganization and redefinition	**Zero application and database downtime** for • Changing initialization parameters dynamically • Renaming and relocating datafiles online • Automatic memory management tuning • Online reorganization and redefinition (managing tables and managing indexes) See the *Oracle Database Administrator Guide*, *Oracle Database Reference* (to evaluate which parameters to use on dynamic), and Online Data Reorganization and Redefinition
Operating system software or hardware updates and patches	**Zero database downtime** with Oracle RAC and Oracle RAC One Node Rolling **Seconds to minutes database downtime** with Standby-First Patch Apply and subsequent Data Guard Switchover
Oracle Database or Grid Infrastructure interim or diagnostic patches	**Zero downtime** with Database Online Patching or Zero-Downtime Oracle Grid Infrastructure Patching **Zero database downtime** with Oracle RAC and Oracle RAC One Node Rolling **Zero application downtime** with Application Checklist for Continuous Service for MAA Solutions **Seconds to minutes database downtime** with Standby-First Patch Apply and subsequent Data Guard Switchover

Table 5-1 (Cont.) Oracle High Availability Solutions for Planned Maintenance

Maintenance Event	High Availability Solutions with Target Outage Time
Oracle Database or Grid Infrastructure quarterly updates under the Critical Patch Update (CPU) program, or Oracle Grid Infrastructure release upgrades	**Zero database downtime** with Oracle RAC and Oracle RAC One Node Rolling. **Zero application downtime** with Application Checklist for Continuous Service for MAA Solutions **Seconds to minutes downtime** with Standby-First Patch Apply and subsequent Data Guard Switchover Special consideration is required during rolling database quarterly updates for applications that use database OJVM. See My Oracle Support RAC Rolling Install Process for the "Oracle JavaVM Component Database PSU/RU" (OJVM PSU/RU) Patches (Doc ID 2217053.1) for details.
Oracle Database Release Upgrade (for example, Oracle Database 11g to 12.2 or 12.2 to 19c)	**Seconds to minutes downtime** with Data Guard transient logical or DBMS_ROLLING solution **Zero downtime** with Oracle GoldenGate See Automated Database Upgrades using Oracle Active Data Guard and DBMS_ROLLING for 12.2 and higher database releases or Database Rolling Upgrade using Data Guard for older releases.
Exadata database server software updates	**Zero database downtime** with Oracle RAC Rolling **Zero application downtime** with Application Checklist for Continuous Service for MAA Solutions **Seconds to minutes downtime** with Standby-First Patch Apply and subsequent Data Guard Switchover See Updating Exadata Software
Exadata storage server or Exadata switch software updates	**Zero downtime** using Exadata `patchmgr` See Updating Exadata Software
Database Server or Oracle RAC cluster changes (add node, drop node, adjust CPU or memory size of the database server)	Some hardware changes like adjusting CPU can be done online without restarting the database server. Refer to the hardware specific documentation. If the change is not online, then **Zero database downtime** with Oracle RAC and Oracle RAC One Node Rolling. **Zero application downtime** with Application Checklist for Continuous Service for MAA Solutions **Seconds to minutes downtime** with Standby-First Patch Apply and subsequent Data Guard Switchover
Application upgrades	**Zero downtime** with Edition Based Redefinition **Zero downtime** with Oracle GoldenGate See Edition Based Redefinition and Oracle GoldenGate documentation

Table 5-1 (Cont.) Oracle High Availability Solutions for Planned Maintenance

Maintenance Event	High Availability Solutions with Target Outage Time
Fleet-wide software maintenance events • Oracle Database or Grid Infrastructure interim or diagnostic patches • Oracle Database or Grid Infrastructure quarterly updates under the Critical Patch Update (CPU) program, or Oracle Grid Infrastructure release upgrades • Exadata database server software updates • Exadata storage server or Exadata switch software updates	Use Fleet Patching and Provisioning, which leverages the following high availability solutions to achieve the target outage times for fleet-wide software maintenance events: **Zero database downtime** with Oracle RAC and Oracle RAC One Node Rolling **Zero application downtime** with Application Checklist for Continuous Service for MAA Solutions for • Oracle Database or Grid Infrastructure interim or diagnostic patches • Oracle Database or Grid Infrastructure quarterly updates under the Critical Patch Update (CPU) program, or Oracle Grid Infrastructure release upgrades • Exadata database server software updates **Zero downtime** using Exadata `patchmgr` for Exadata storage server or Exadata switch software updates

High Availability Solutions for Migration

Oracle MAA recommends several solutions for reducing downtime due to database migration.

The following table describes the high availability solutions for migration at a high level.

Table 5-2 High Availability Solutions for Migration

Maintenance Event	High Availability Solutions with Target Outage Time
Migrate to an on-premises Oracle Exadata Database Machine or any Oracle Database cloud service, including Oracle Exadata Database Service on Cloud@Customer See Zero Downtime Migration for a complete list of supported services and platforms	Use the Zero Downtime Migration tool, which provides • Physical migration with RMAN backup and restore, with an optional low downtime option using Oracle Data Guard. This is the simplest turnkey migration solution, which is ideal when the source and target system platform (for example, Linux to Linux) and database versions (Oracle Database 19c to 19c) are the same. • Logical migration with Oracle Data Pump, with an optional low downtime option using Oracle GoldenGate. This is the only option for migrating a database when the source and target system platform (For example, AIX to Linux) or major database versions (Oracle Database 12c to 19c) are different. To migrate to Oracle Autonomous Database, use the Oracle Cloud Infrastructure Database Migration service (or the Zero Downtime Migration tool), which provide • Offline migration with Data Pump • Online migration with Data Pump and Oracle GoldenGate

Table 5-2 (Cont.) High Availability Solutions for Migration

Maintenance Event	High Availability Solutions with Target Outage Time
Migrate the database to a different server or platform	**Seconds to minutes downtime** with Oracle Data Guard for certain platform combinations **Zero downtime** with Oracle GoldenGate Data Guard always supports primary and standby combinations on the same platform. For heterogeneous platforms, Refer to Data Guard Support for Heterogeneous Primary and Physical Standbys in Same Data Guard Configuration (Doc ID 413484.1)
Migrate database to an incompatible character set	**Zero downtime** with Oracle GoldenGate See Character Set Migration
Migrate to pluggable databases to another container database	**Seconds to minutes downtime** with Pluggable Database Relocate (PDB Relocate) See Relocating a PDB
Migrate to new storage	**Zero Downtime** with Oracle Automatic Storage Management if storage is compatible with Oracle Data Guard for certain platform combinations **Zero Downtime** with Oracle GoldenGate
Migrate database from a single-instance system to an Oracle RAC cluster	**Zero Downtime** with Oracle RAC when applicable. See Adding Oracle RAC to Nodes with Oracle Clusterware Installed **Seconds to minutes downtime** with Oracle Data Guard for certain platform combinations **Zero Downtime** with Oracle GoldenGate

6

Enabling Continuous Service for Applications

Applications achieve continuous service easily when the underlying network, systems, and databases are always available.

To achieve continuous service in the face of unplanned outages and planned maintenance activities can be challenging. An MAA database architecture and its configuration and operational best practices is built upon redundancy and its ability to tolerate, prevent, and at times auto-repair failures.

However, applications can incur downtime whenever a failure hits a database instance, a database node, or the entire cluster or data center. Similarly, some planned maintenance activities may require restarting a database instance, a database node, or an entire database server to be restarted.

In all cases, following a simple checklist, your applications can incur zero or very little downtime whenever the database service that the application is connected to can be moved to another Oracle RAC instance or to another database.

See Configuring Continuous Availability for Applications for various levels and options to achieve continuous service for your application.

Drain Timeouts for Planned Maintenance Events

For planned maintenance events, some applications require time to complete their in-flight transactions.

The amount of time (`DRAIN_TIMEOUT`) for any workload to gracefully complete its in-flight transactions and move its sessions vary based on the workload characteristics. For short OLTP transactions, a `DRAIN_TIMEOUT` of 1 minute may be sufficient, while batch jobs might require 30 minutes. In some cases it might be best to suspend these long transactions to times outside the planned maintenance window.

The trade-off for configuring a longer `DRAIN_TIMEOUT` is that the planned maintenance window would be extended.

The following table outlines planned maintenance events that will incur Oracle RAC instance rolling restart and the relevant service drain timeout variables that may impact your application.

Table 6-1 Drain Timeout Variables for Planned Maintenance Events

Planned Maintenance Event	Application Drain Timeout Variables
Exadata Database Host (Dom0) software changes	Exadata Host handles operating system (OS) shutdown with maximum timeout of 10 minutes. OS shutdown calls an `rhphelper`, which has the following drain timeout settings: • `DRAIN_TIMEOUT`: value used for services that do not have a `drain_timeout` defined. Default 180 • `MAX_DRAIN_TIMEOUT`: overrides any higher `drain_timeout` value defined for a given service. Default 300 Each Clusterware-managed service is also controlled by a `drain_timeout` attribute that can be lower than the above values. See also: Using RHPhelper to Minimize Downtime During Planned Maintenance on Exadata (Doc ID 2385790.1)
Exadata Database Guest (DomU) software changes	Exadata `patchmgr` and `dbnodeupdate` software programs call `rhphelper`, which has the following drain timeout settings: `DRAIN_TIMEOUT`: value used for services that do not have a `drain_timeout` defined. Default 180 `MAX_DRAIN_TIMEOUT`: overrides any higher `drain_timeout` value defined for a given service. Default 300 Each Clusterware-managed service is also controlled by a `drain_timeout` attribute that can be lower than the above values. See also: Using RHPhelper to Minimize Downtime During Planned Maintenance on Exadata (Doc ID 2385790.1)
Oracle Grid Infrastructure (GI) software changes or upgrade	The recommend steps are described in Graceful Application Switchover in RAC with No Application Interruption (Doc ID 1593712.1). Example: `srvctl stop instance -o immediate -drain_timeout 600 -failover -force` Each Clusterware-managed service is also controlled by a `drain_timeout` attribute that can be lower than the above values.
Oracle Database Software changes	The recommend steps are described in Graceful Application Switchover in RAC with No Application Interruption (Doc ID 1593712.1). Example: `srvctl stop instance -o immediate -drain_timeout 600 -failover -force` Each Clusterware-managed service is also controlled by a `drain_timeout` attribute that can be lower than the above values.

See also:

Configuring Continuous Availability for Applications

7

Operational Prerequisites to Maximizing Availability

Use the following operational best practices to provide a successful MAA implementation.

Understand High Availability and Performance Service-Level Agreements

Understand and document your high availability (HA) and performance service-level agreements (SLAs):

- Understand the attributes of High Availability and various causes of downtime as described in Overview of High Availability.
- Establish high availability and performance SLAs from lines of business, upper management, and technical teams as described in High Availability Requirements, and A Methodology for Documenting High Availability Requirements.

Implement and Validate a High Availability Architecture That Meets Your SLAs

When you have agreement on your high availability and performance service level requirements:

- **Map** the requirements to one of the Oracle MAA standard and validated MAA reference architectures, as described in High Availability and Data Protection – Getting From Requirements to Architecture
- **Evaluate** the outage and planned maintenance matrices relevant to your reference architecture in Oracle Database High Availability Solutions for Unplanned Downtime and Oracle Database High Availability Solutions for Planned Downtime
- **Learn** about the database features required to implement your MAA architecture in High Availability Architectures

Establish Test Practices and Environment

You must **validate** or **automate** the following to ensure that your high availability SLAs are met:

- All software update and upgrade maintenance events
- All repair operations, including those for various types of unplanned outages
- Backup, restore, and recovery operations

If you use Oracle Data Guard for disaster recovery and data protection, Oracle recommends that you:

- Perform periodic switchover operations, or conduct full application and database failover tests
- Validate end-to-end role transition procedures by performing application and Data Guard switchovers periodically

A good test environment and proper test practices are essential prerequisites to achieving the highest stability and availability in your production environment. By validating every change in your test environment thoroughly, you can proactively detect, prevent, and avoid problems before applying the same change on your production systems.

These practices involve the following:

Configuring Test and QA Environments

The test environment should be a replica of the production MAA environment (for example, using the MAA Gold reference architecture.) There will be trade offs if the test system is not identical to the MAA service-level driven standard reference architecture that you plan to implement. It is recommended that you perform functional, performance, and availability tests with a workload that mimics production. Evaluate if availability and performance SLAs are maintained after each change, and ensure that clear fallback or repair procedures are in place if things go awry, while applying the change on the production environment.

With a properly configured test system, many problems can be avoided, because changes are validated with an equivalent production and standby database configuration containing a full data set and using a workload framework to mimic production (for example, using Oracle Real Application Testing.)

Do not try to reduce costs by eliminating the test system, because that decision ultimately affects the stability and the availability of your production applications. Using only a subset of system resources for testing and QA has the tradeoffs shown in the following table, which is an example of the MAA Gold reference architecture.

Table 7-1 Tradeoffs for Different Test and QA Environments

Test Environment	Benefits and Tradeoffs
Full Replica of Production and Standby Systems	**Validate:** • All software updates and upgrades • All functional tests • Full performance at production scale • Full high availability and disaster recovery testing
Full Replica of Production Systems	**Validate:** • All software updates and upgrades • All functional tests • Full performance at production scale • Full high availability minus the standby system **Cannot Validate:** • Disaster recovery testing • Any standby redo apply or read only workload performance testing • Redo transport performance and impact on production system resources due to redo transport • Any use case using the standby database such as Database Rolling Upgrade, Snapshot Standby, and so on.

Table 7-1 (Cont.) Tradeoffs for Different Test and QA Environments

Test Environment	Benefits and Tradeoffs
Standby System	**Validate:** • Most software update changes • All read-only functional tests • Full performance--if using Data Guard Snapshot Standby, but this can extend recovery time if a failover is required • Resource management and scheduling--required if standby and test databases exist on the same system **Cannot Validate:** • Role transition and disaster recovery testing • Any use case using the standby database such as Database Rolling Upgrade, Snapshot Standby, and so on.
Shared System Resource	**Validate:** • Most software update changes • All functional tests **Cannot Validate:** This environment may be suitable for performance testing if enough system resources can be allocated to mimic production. Typically, however, the environment includes a subset of production system resources, compromising performance validation. Resource management and scheduling is required. Standby or disaster recovery testing may not be possible or limited.
Smaller or Subset of the system resources	**Validate:** • All software update changes • All functional tests • Limited full-scale high availability evaluations **Cannot Validate:** • Performance testing at production scale • Standby or disaster recovery testing may not be possible or limited.
Different hardware or platform system resources but same operating system	**Validate:** • Some software update changes • Limited firmware patching test • All functional tests unless limited by new hardware features • Limited production scale performance tests • Limited full-scale high availability evaluations • Standby or disaster recovery testing may not be possible or limited.

See Also:

Oracle Database Testing Guide

Performing Preproduction Validation Steps

Pre-production validation and testing of hardware, software, database, application or any changes is an important way to maintain stability. The high-level pre-production validation steps are:

1. Review the patch or upgrade documentation or any document relevant to that change. Evaluate the possibility of performing a rolling upgrade if your SLAs require zero or minimal downtime. Evaluate any rolling upgrade opportunities to minimize or eliminate planned downtime. Evaluate whether the patch or the change qualifies for Standby-First Patching.

   Note:

   Standby-First Patch enables you to apply a patch initially to a physical standby database while the primary database remains at the previous software release (this applies to certain types of software updates and does not apply to major release upgrades; use the Data Guard transient logical standby and DBMS_ROLLING method for patch sets and major releases). Once you are satisfied with the change, then perform a switchover to the standby database. The fallback is to switchback if required. Alternatively, you can proceed to the following step and apply the change to your production environment. For more information, see "Oracle Patch Assurance - Data Guard Standby-First Patch Apply" in My Oracle Support Note 1265700.1 at https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=1265700.1

2. Validate the application in a test environment and ensure the change meets or exceeds your functionality, performance, and availability requirements. Automate the procedure and be sure to also document and test a fallback procedure. This requires comparing metrics captured before and after patch application on the test and against metrics captured on the production system. Real Application Testing may be used to capture the workload on the production system and replay it on the test system. AWR and SQL Performance Analyzer may be used to assess performance improvement or regression resulting from the patch.

   Validate the new software on a test system that mimics your production environment, and ensure the change meets or exceeds your functionality, performance, and availability requirements. Automate the patch or upgrade procedure and ensure fallback. Being thorough during this step eliminates most critical issues during and after the patch or upgrade.

3. Use Oracle Real Application Testing and test data management features to comprehensively validate your application while also complying with any security restrictions your line of business may have. Oracle Real Application Testing (a separate database option) enables you to perform real-world testing of Oracle Database. By capturing production workloads and assessing the impact of system changes on these workloads before production deployment, Oracle Real Application Testing minimizes the risk of instabilities associated with system changes. SQL Performance Analyzer and Database Replay are key components of Oracle Real Application Testing. Depending on the nature and impact of the system change being tested, and on the type of system on which the test will be performed, you can use either or both components to perform your testing.

   When performing real-world testing there is a risk of exposing sensitive data to non-production users in a test environment. The test data management features of Oracle Database help to minimize this risk by enabling you to perform data masking and data subsetting on the test data.

4. If applicable, perform final pre-production validation of all changes on a Data Guard standby database before applying them to production. Apply the change in a Data Guard environment, if applicable.

5. Apply the change in your production environment.

> **See Also:**
>
> Data Guard Redo Apply and Standby-First Patching and Data Guard Transient Logical Rolling Upgrades
>
> Converting a Physical Standby Database into a Snapshot Standby Database and Performing a Rolling Upgrade With an Existing Physical Standby Database in Oracle Data Guard Concepts and Administration
>
> Oracle Database Rolling Upgrades: Using a Data Guard Physical Standby Database on http://www.oracle.com/goto/maa
>
> Oracle Patch Assurance - Data Guard Standby-First Patch Apply (Doc ID 1265700.1)

Set Up and Use Security Best Practices

Corporate data can be at grave risk if placed on a system or database that does not have proper security measures in place. A well-defined security policy can help protect your systems from unwanted access and protect sensitive corporate information from sabotage. Proper data protection reduces the chance of outages due to security breaches.

> **See Also:**
>
> *Oracle Database Security Guide*.

Establish Change Control Procedures

Institute procedures that manage and control changes as a way to maintain the stability of the system and to ensure that no changes are incorporated in the primary database unless they have been rigorously evaluated on your test systems, or any one of the base architectures in the MAA service-level tiers.

Review the changes and get feedback and approval from your change management team.

Apply Recommended Software Updates and Security Updates Periodically

Maintaining software at current or recent versions provides many benefits, such as better software security, improved resource utilization and stability, continued compatibility with newer related software, better support and faster resolution of issues, and the ability to receive fixes for newly discovered issues.

Update all software on a regular basis. Oracle recommends following these practices:

- Learn the release and support timelines for all software that your MAA environment depends upon in order to develop a plan for upgrade to a new major software release and a plan for installing proactive updates for current releases.

  For example, Oracle Database release and support timelines is available in My Oracle Support Note 742060.1 “Release Schedule of Current Database Releases”.

- Upgrade to a later major software release before proactive software updates for your current release cease.
- Install proactive software updates for your current release as they become available, typically on a monthly or quarterly basis.

  However, business requirements may dictate that the adoption of certain proactive updates is delayed or skipped. In such cases Oracle recommends that the currently running software never lags the most recently released proactive update by more than 12 months.
- Install reactive software patches (also known as interim or one-off patches) for critical issues published in My Oracle Support Alerts as soon as feasible.
- Validate the software update process and perform soak testing on a test system before updating software on production systems.
- Use Oracle health check tools, Orachk and Exachk, to provide Oracle software upgrade and proactive update advice, critical issue software update recommendations, patching and upgrading pre-checks, database and system health checks, and MAA recommendations.

  Orachk supports non-engineered systems and Oracle Database Appliance. Exachk supports engineered systems Oracle Exadata Database Machine and Oracle Zero Data Loss Recovery Appliance.

See also:

For Oracle Database and Grid Infrastructure:

- “Release Schedule of Current Database Releases” in My Oracle Support Note 742060.1
- "Primary Note for Database Proactive Patch Program" in My Oracle Support Note 888.1
- "Oracle Database 19c Important Recommended One-off Patches" in My Oracle Support Note 555.1

For engineered systems (Exadata Database Machine and Zero Data Loss Recovery Appliance):

- "Exadata Database Machine and Exadata Storage Server Supported Versions" in My Oracle Support Note 888828.1
- “Exadata Critical Issues” in My Oracle Support Note 1270094.1
- "Oracle Exadata: Exadata and Linux Important Recommended Fixes" in My Oracle Support Note 556.1
- "Oracle Exadata Database Machine Exachk" in My Oracle Support Note 1070954.1

For non-engineered systems:

- "Autonomous Health Framework (AHF) - Including TFA and Orachk/Exachk" in My Oracle Support Note 2550798.1

Establish Disaster Recovery Environment

To achieve the same performance and HA characteristics as the source or primary database, the disaster recovery environment or target should be symmetric or similarly configured to the production system.

If the disaster recovery target is a standby database or Oracle GoldenGate replica, symmetric or similar database compute CPU, memory, and throughput is required to match the same performance. Similarly, the storage should be able to handle the same IOPS, throughput, and response time.

When the disaster recovery target is used by other applications or databases for database consolidation and cost efficiency, additional resources will be required to ensure acceptable performance with other concurrent workloads.

Establish and Validate Disaster Recovery Practices

Disaster recovery validation is required to ensure that you meet your disaster recovery service level requirements such as RTO and RPO.

Whether you have a standby database, Oracle GoldenGate replica, or leverage database backups from Zero Data Loss Recovery Appliance (Recovery Appliance), ZFS Storage, or another third party, it is important to ensure that the operations and database administration teams are well prepared to failover or restore the database and application any time the primary database is down or underperforming. The concerned teams should be able to detect and decide to failover or restore as required. Such efficient preparation before disasters will significantly reduce overall downtime.

If you use Data Guard or Oracle GoldenGate for high availability, disaster recovery, and data protection, Oracle recommends that you perform regular application and database switchover operations **every three to six months,** or conduct full application and database failover tests.

Periodic RMAN cross checks, RMAN backup validations, and complete database restore and recovery are required to validate your disaster recovery solution through backups. Inherent backup checks and validations are done automatically with the Recovery Appliance, but periodic restore and recovery tests are still recommended.

See also: Role Transition, Assessment, and Tuning

Establish Escalation Management Procedures

Establish escalation management procedures so repair is not hindered. Most repair solutions, when conducted properly are automatic and transparent with the MAA solution. The challenges occur when the primary database or system is not meeting availability or performance SLAs and failover procedures are not automatic as in the case with some Data Guard failover scenarios. Downtime can be prolonged if proper escalation policies are not followed and decisions are not made quickly.

If availability is the top priority, perform failover and repair operations first and then proceed with gathering logs and information for Root Cause Analysis (RCA) after the application service has been reestablished. For simple data gathering, use the Trace File Analyzer Collector (TFA).

> **See Also:**
>
> MAA web page at `http://www.oracle.com/goto/maa`
>
> My Oracle Support note 1513912.2 “TFA Collector - Tool for Enhanced Diagnostic Gathering” at 1513912.2

Configure Monitoring and Service Request Infrastructure for High Availability

To maintain your High Availability environment, you should configure the monitoring infrastructure that can detect and react to performance and high availability related thresholds before any downtime has occurred.

Also, where available, Oracle can detect failures, dispatch field engineers, and replace failed hardware components such as disks, flash cards, fans, or power supplies without customer involvement.

Run Database Health Checks Periodically

Oracle Database health checks are designed to evaluate your hardware and software configuration and MAA compliance to best practices.

All of the Oracle health check tools will evaluate Oracle Grid Infrastructure, Oracle Database, and provide an automated MAA scorecard or review that highlights when key architectural and configuration settings are not enabled for tolerance of failures or fast recovery. For Oracle's engineered systems such as Exadata Database Machine, there may be hundreds of additional software, fault and configuration checks.

Oracle recommends periodically (for example, monthly for Exadata Database Machine) downloading the latest database health check, running the health check, and addressing the key FAILURES, WARNINGS, and INFO messages. Use Exachk for Engineered Systems such as Oracle Exadata Database Machine and Oracle Zero Data Loss Recovery Appliance, and use Orachk for non-engineered systems and Oracle Database Appliance.

Furthermore, it is recommended that you run the health check prior to and after any planned maintenance activity.

You must evaluate:

- Existing or new critical health check alerts prior to planned maintenance window
- Existing software or critical software recommendations
- Adding any new recommendations to the planned maintenance window after testing

> **See Also:**
>
> My Oracle Support Note 1268927.2 "ORAchk - Health Checks for the Oracle Stack" at https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=1268927.2
>
> My Oracle Support Note 1070954.1 "Oracle Exadata Database Machine exachk or HealthCheck" at https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=1070954.1

Configure Monitoring

When deciding on the best route for monitoring your Exadata fleet, you need to consider how the fleet you are monitoring is deployed (On-Premises, Cloud@Customer, Oracle Cloud Infrastructure) and where your monitoring is or can be deployed.

- **On-Premises**

  For fleets including on-premises Exadata, Enterprise Manager includes necessary monitoring for responsibilities spanning all three deployment types and is the MAA Best Practice.

- **Cloud**

  For fleets only in Cloud@Customer and/or OCI, who do not currently have Enterprise Manager or On-Premises monitoring deployment options, the OCI Observability & Management services provide various options for basic and advanced monitoring and manageability.

Configure Oracle Enterprise Manager Monitoring

If your Exadata fleet includes On-Premises deployment, you should configure and use Enterprise Manager and the monitoring infrastructure that detects and reacts to performance and high availability related thresholds to avoid potential downtime.

The monitoring infrastructure assists you with monitoring for High Availability and enables you to do the following:

- Monitor system, network, application, database and storage statistics
- Monitor performance and service statistics
- Create performance and high availability thresholds as early warning indicators of system or application problems
- Provide performance and availability advice
- Established alerts and tools and database performance
- Receive alerts for engineered systems hardware faults

Enterprise Manager provides monitoring and management for Exadata and Databases deployed on-premises, on Cloud@Customer, and OCI.

- For on-premises Exadata deployments, see Oracle Exadata Database Machine Getting Started Guide.
- For Cloud@Customer and OCI Exadata deployments, see Oracle Enterprise Manager Cloud Control for Oracle Exadata Cloud

Configure Enterprise Manager for high availability to ensure that the manageability solution is as highly available as the systems that you're monitoring.

For configuration details for HA see Oracle Enterprise Manager Cloud Control Advanced Installation and Configuration Guide. For additional MAA Best Practices for Enterprise Manager see http://www.oracle.com/goto/maa**.**

Oracle Observability and Management Services can be used in conjunction with Enterprise Manager to provide additional Exadata manageability features. For details, see the following:

- Operations Insights Exadata Insights
- Harvest Entity Model Data from Enterprise Manager Cloud Control Collect Logs

Configure OCI Observability and Management Services Monitoring

If your Exadata fleet includes only Cloud@Customer and/or OCI deployment, and you do not currently have Enterprise Manager or on-premises monitoring deployment options, you should configure and use the OCI Observability and Management platform of services that work together to provide monitoring and management of Oracle Cloud targets.

Basic default metrics and events for performance, high availability, and health are available in the OCI console. For details see the following documentation:

- Exadata Database Service on Dedicated Infrastructure
  - Metrics for VM Clusters and Exadata Database Service on Dedicated Infrastructure available in the Monitoring Service
  - Events
- Exadata Database Service on Cloud@Customer
  - Metrics for VM Clusters and Exadata Database Service on Cloud@Customer available in the Monitoring Service
  - Events

Advanced metrics and management features are available in the Database Management service:

- Exadata Database Service on Dedicated Infrastructure
  - Metrics available in the Database Management Service
  - Diagnose and Troubleshoot Problems with Pluggable Databases

Advanced analytics features are available in the Operations Insights Service:

- Exadata Database Service on Dedicated Infrastructure
  - Operations Insights for Oracle Databases

See also: Oracle Cloud Observability and Management Platform

Configure Automatic Service Request Infrastructure

In addition to monitoring infrastructure with Enterprise Manager, Oracle can detect failures, dispatch field engineers, and replace failing hardware without customer involvement.

For example, Oracle Automatic Service Request (ASR) is a secure, scalable, customer-installable software solution available as a feature. The software resolves problems faster by using auto-case generation for Oracle's server and storage systems when specific hardware faults occur.

See also: Oracle Automatic Service Request (Doc ID 1185493.1)

Exercise Capacity Planning

Periodically perform capacity planning exercises to ensure that your current hardware resources can accommodate existing workload and projected growth.

With database consolidation, this exercise should be done before migrating or adding a new database to the existing system.

Note that concurrent workloads can interfere with each other and can cause unpredictable behavior at times, so performance and HA testing may be required.

Using Database multitenant container databases, database resource management, or Exadata consolidation practices can help optimize existing system resources and constrain workload usage to meet expectations.

Check the Latest MAA Best Practices

The MAA solution encompasses the full stack of Oracle technologies, so you can find MAA best practices for Oracle Database, Oracle Cloud, Oracle Exadata, Zero Data Loss Recovery Appliance, Oracle Fusion Middleware, Oracle Applications Unlimited, and Oracle Enterprise Manager on the MAA pages.

MAA solutions and best practices continue to be developed and published on http://www.oracle.com/goto/maa.

Part II

Oracle Database High Availability Best Practices

8

Overview of Oracle Database High Availability Best Practices

By adopting the Oracle MAA best practices for Oracle Database, you can achieve the service levels of the Oracle MAA Bronze reference architecture.

The Bronze architecture achieves the highest availability for a single-instance database configuration, whether it is a standalone database or part of a consolidated multitenant database, by using the high availability capabilities included in Oracle Database Enterprise Edition.

The Bronze architecture is the base configuration for the other MAA reference architectures. The Oracle Database best practices should also be implemented in the Silver, Gold, and Platinum references architectures, unless specifically noted in the best practices for that architecture.

For information about the components, service levels, and benefits of the Bronze reference architecture, as well as the MAA architectures that build on the Bronze base, see the "High Availability Reference Architectures" interactive diagram at https://www.oracle.com/webfolder/technetwork/tutorials/architecture-diagrams/high-availability-overview/high-availability-reference-architectures.html.

9

Oracle Database Configuration Best Practices

Adopt the Oracle MAA best practices for configuring all Oracle single-instance databases to reduce or avoid outages, reduce the risk of corruption, and improve recovery performance.

Note that the following Oracle Database best practices are used to configure the Oracle MAA Bronze reference architecture, and they are also the base database base practices for the other MAA reference architectures: Silver (Oracle RAC), Gold (Oracle Data Guard), and Platinum (Oracle GoldenGate).

Use a Server Parameter File (SPFILE)

The server parameter file (SPFILE) enables a single, central parameter file to hold all database initialization parameters associated with all instances of a database. This provides a simple, persistent, and robust environment for managing database parameters. SPFILE is recommended to be placed in the DATA ASM disk group.

Enable Archive Log Mode and Forced Logging

Running the database in `ARCHIVELOG` mode and using database `FORCE LOGGING` mode are prerequisites for database recovery operations.

The `ARCHIVELOG` mode enables online database backup and is necessary to recover the database to a point in time later than what has been restored. Features such as Oracle Data Guard and Flashback Database require that the production database run in `ARCHIVELOG` mode.

If you can isolate data that never needs to be recovered within specific tablespaces, then you can use tablespace level `FORCE LOGGING` attributes instead of the database `FORCE LOGGING` mode.

Configure an Alternate Local Archiving Destination

The local archive destination, usually `LOG_ARCHIVE_DEST_1`, should have an alternate local destination on a different ASM disk group. This configuration prevents database hanging due to lack of archive log space if `DB_RECOVERY_FILE_DEST` fills up or is unavailable for any reason.

Table 9-1 Alternate Local Archiving Configuration Parameters

Database Parameter	LOG_ARCHIVE_DEST_n parameter settings for local archive destinations
LOG_ARCHIVE_DEST_*n*	LOCATION=USE_DB_FILE_RECOVERY_DEST VALID_FOR=(ALL_LOGFILES,ALL_ROLES) MAX_FAILURE=1 REOPEN=5 DB_UNIQUE_NAME=db_unique_name of the database ALTERNATE=some other log archive destination. Must be log_archive_dest_[1-10]
LOG_ARCHIVE_DEST_*y*	LOCATION=A disk group other than the disk group used for DB_RECOVERY_FILE_DEST. Usually the DATA disk group. VALID_FOR=(ALL_LOGFILES,ALL_ROLES) MAX_FAILURE=1 REOPEN=5 ALTERNATE= the primary local archive log destination: usually LOG_ARCHIVE_DEST_1
DB_RECOVERY_FILE_DEST	Archive destination, for example, a RECO disk group
LOG_ARCHIVE_DEST_STATE_*n*	ENABLE
LOG_ARCHIVE_DEST_STATE_*y*	ALTERNATE

Sample parameter settings:

- LOG_ARCHIVE_DEST_1='LOCATION=USE_DB_FILE_RECOVERY_DEST VALID_FOR=(ALL_LOGFILES,ALL_ROLES) MAX_FAILURE=1 REOPEN=5 DB_UNIQUE_NAME=*db_unique_name of the database* ALTERNATE=LOG_ARCHIVE_DEST_10'
- LOG_ARCHIVE_DEST_10='LOCATION=+DATA VALID_FOR=(ALL_LOGFILES,ALL_ROLES) MAX_FAILURE=1 REOPEN=5 DB_UNIQUE_NAME=*db_unique_name of the database* ALTERNATE=LOG_ARCHIVE_DEST_1'
- LOG_ARCHIVE_DEST_STATE_1 =enable
- LOG_ARCHIVE_DEST_STATE_10=alternate
- DB_RECOVERY_FILE_DEST=*typically the RECO disk group*

Use a Fast Recovery Area

The Fast Recovery Area is Oracle-managed disk space that provides a centralized disk location for backup and recovery files.

The Fast Recovery Area is defined by setting the following database initialization parameters:

- DB_RECOVERY_FILE_DEST specifies the default location for the fast recovery area. Set this parameter to the RECO disk group.
- DB_RECOVERY_FILE_DEST_SIZE specifies (in bytes) the hard limit on the total space to be used by database recovery files created in the recovery area location.

Set this parameter to a value large enough to store archived logs, flashback logs and any local database backup files locally. Having the files locally can reduce your recovery time after restoring a backup. RMAN will automatically manage these files according to your RMAN backup and data retention policies. Typically customers store 24 hours of data in the destination

When your system hosts many databases sharing the same `DB_RECOVERY_FILE_DEST_SIZE`, space needs to manage and monitored holistically. Recommended to alert when RECO disk group for example is 90% full.

Enable Flashback Database

Flashback Database provides an efficient alternative to point-in-time recovery for reversing unwanted database changes.

Flashback Database lets you rewind an entire database backward in time, reversing the effects of database changes within a time window. The effects are similar to database point-in-time recovery. You can flash back a database by running a single RMAN command or a SQL*Plus statement instead of using a complex procedure.

To enable Flashback Database, configure a fast recovery area and set a flashback retention target using the best practices listed below. This retention target specifies how far back you can rewind a database with Flashback Database.

- Know your application performance baseline before you enable flashback database to help determine the overhead and to assess the application workload implications of enabling flashback database.
- Ensure that the fast recovery area space is sufficient to hold the flashback database flashback logs. A general rule of thumb is that the volume of flashback log generation is approximately the same order of magnitude as redo log generation. For example, if you intend to set `DB_FLASHBACK_RETENTION_TARGET` to 24 hours, and if the database generates 20 GB of redo in a day, then allow 20 GB to 30 GB disk space for the flashback logs.
  - An additional method to determine fast recovery area sizing is to enable flashback database and allow the database to run for a short period of time (2-3 hours). Query `V$FLASHBACK_DATABASE_STAT.ESTIMATED_FLASHBACK_SIZE` to retrieve the estimated amount of space required for the fast recovery area.
  - Note that the `DB_FLASHBACK_RETENTION_TARGET` is a target and there is no guarantee that you can flashback the database that far. In some cases if there is space pressure in the fast recovery area where the flashback logs are stored, then the oldest flashback logs may be deleted. To guarantee a flashback point-in-time you must use guaranteed restore points.
- Ensure that there is sufficient I/O bandwidth to the fast recovery area. Insufficient I/O bandwidth with flashback database on is usually indicated by a high occurrence of the `FLASHBACK BUF FREE BY RVWR` wait event.
- To monitor the progress of a flashback database operation you can query the `V$SESSION_LONGOPS` view. An example query to monitor progress is

  ```
  SELECT sofar, totalwork, units FROM v$session_longops WHERE opname =
  'Flashback Database';
  ```

- For repetitive tests where you must flashback to the same point, use flashback database guaranteed restore points instead of enabling flashback database. This will minimize space usage.

- Flashback PDB can rewind a pluggable database without affecting other PDBs in the CDB. You can also create PDB restore points.

Set FAST_START_MTTR_TARGET Initialization Parameter

With Fast-Start Fault Recovery, the `FAST_START_MTTR_TARGET` initialization parameter simplifies the configuration of recovery time from instance or system failure.

The `FAST_START_MTTR_TARGET` parameter specifies a target for the expected recovery time objective (RTO), which is the time, in seconds, that it should take to start the instance and perform cache recovery. When you set this parameter, the database manages incremental checkpoint writes in an attempt to meet the target. If you have chosen a practical value for this parameter, then you can expect your database to recover, on average, in approximately the number of seconds you have chosen.

Initially, set the `FAST_START_MTTR_TARGET` initialization parameter to 300 (seconds), or to the value required for your expected recovery time objective (RTO). As you set or lower this value, database writer (DBWR) will become more active to meet your recovery targets.

Make sure that you have sufficient IO bandwidth to handle potential higher load. See the Database Performance Tuning Guide for information about monitoring and tuning `FAST_START_MTTR_TARGET`.

Outage testing for cases such as node or instance failures during peak loads is recommended.

Protect Against Data Corruption

Oracle Database corruption prevention, detection, and repair capabilities are built on internal knowledge of the data and transactions it protects, and on the intelligent integration of its comprehensive high availability solutions.

A data block is corrupted when it is not in a recognized Oracle Database format, or its contents are not internally consistent. Data block corruption can damage internal Oracle control information or application and user data, leading to crippling loss of critical data and services.

When Oracle Database detects corruption, it offers block media recovery and data file media recovery to recover the data. You can undo database-wide logical corruptions caused by human or application errors with Oracle Flashback Technologies. Tools are also available for proactive validation of logical data structures. For example, the SQL*Plus `ANALYZE TABLE` statement detects inter-block corruptions.

The following are best practices for protecting your database against corruption.

- Use Oracle Automatic Storage Management (Oracle ASM) to provide disk mirroring to protect against disk failures.
- Use the `HIGH` redundancy disk type for optimal corruption repair with Oracle ASM.

  Using Oracle ASM redundancy for disk groups provides mirrored extents that can be used by the database if an I/O error or corruption is encountered. For continued protection, Oracle ASM redundancy lets you move an extent to a different area on a disk if an I/O error occurs. The Oracle ASM redundancy mechanism is useful if you have bad sectors returning media errors.
- Enable Flashback technologies for fast point-in-time recovery from logical corruptions that are most often caused by human error, and for fast reinstatement of a primary database following failover.

- Implement a backup and recovery strategy with Recovery Manager (RMAN) and periodically use the RMAN `BACKUP VALIDATE CHECK LOGICAL` scan to detect corruptions.

  Use RMAN and Oracle Secure Backup for additional block checks during backup and restore operations. Use Zero Data Loss Recovery Appliance for backup and recovery validation including corruption checks and repairs, central backup validation, reduced production database impact, and Enterprise Cloud backup and recovery solutions.
- Set database initialization parameter `DB_BLOCK_CHECKSUM=MEDIUM` or `FULL`.
- Evaluate setting `DB_BLOCK_CHECKING=MEDIUM` or `FULL`, but only after a full performance evaluation with the application.

Set the LOG_BUFFER Initialization Parameter to 128MB or Higher

Set the `LOG_BUFFER` initialization parameter to a minimum of 128 MB for databases with flashback enabled.

Set USE_LARGE_PAGES=ONLY

On Linux, the database's SGA should leverage large pages for consistent performance and stability.

There are two ways to ensure this happens with the `USE_LARGE_PAGES` parameter:

- `USE_LARGE_PAGES=ONLY` - Hugepages must be preallocated before instance startup.
- `USE_LARGE_PAGES=AUTO_ONLY` - Hugepages are dynamically acquired at instance startup time, but this dynamic acquisition can fail if memory is fragmented or if another instance is starting up and dynamically acquiring hugepages at the same time.

The MAA best practice is `USE_LARGE_PAGES=ONLY`. This recommendation is applicable for Cloud and non-Cloud environments, and all Cloud and Exadata automation tools ensure this configuration is in place.

> **Note:**
>
> Oracle RDBMS 19c default for `USE_LARGE_PAGES` on Exadata is `AUTO_ONLY`, but this value will be deprecated in the future.

Use Bigfile Tablespace

As databases grow larger more data files are added to smallfile tablespaces, which requires additional administration, monitoring, and maintenance, while negatively impacting database open time and role transition time in Oracle Data Guard environments.

Bigfile tablespaces allow a single large data file per tablespace, up to 32TB for 8k blocksize and 128TB for 32k blocksize. The single data file reduces the number of files in the database thus improving database checkpoint, database open, and role transition time, while improving administration costs.

Recommendations include:

- For **new database** design and deployment, use bigfile tablespaces and partitioning to minimize the number of data files. Partitioning of large tables prevents having an enormous bigfile. A reasonable bigfile should still be 16TB or less.
  - For very large tables that have different retention policies, or have different access requirements, use Oracle Partitioning as part of your database and object design. Oracle Partitioning can also work around any potential bigfile size limitation.
  - For very large tablespaces, use bigfile tablespaces instead of many smallfile data files. Bigfile tablespaces are only supported for locally managed tablespaces with automatic segment space management.
  - There are no negative trade-offs for using bigfile tablespaces, other than understanding the maximum limits for your `DB_BLOCK_SIZE`. To continue to ensure good database backup and restore performance, you should also use the `RMAN SECTION SIZE` parameter to parallelize backup and restore operations when there are bigfile tablespaces.
- For **existing databases** with a lot of data files, focus on tablespaces that have the most data files and evaluate if you can use the `ALTER TABLE MOVE` or online redefinition to migrate tables or partitions to bigfile tablespaces.

The following tables show a recent Data Guard performance test which demonstrates that reducing the number of data files in the database from 9000 data files to ~100 data files improved failover times by 10x and switchover times by 4 times.

Unplanned Outage/DR (Failover)	**Initial Configuration**	**Tuned MAA Configuration**
Close to Mount (C2M)	21 secs	1 sec
Terminal Recovery (TR)	154 secs	2 secs
Convert to Primary (C2P)	114 secs	5 secs
Open new Primary (OnP)	98 secs	28 secs
Open PDB and Start Service (OPDB)	146 secs	16 secs
Total App Downtime	**533 secs or 8min 53 secs**	**52 secs (90% drop)**

Planned DR Switch (Switchover)	**Initial Configuration**	**Tuned MAA Configuration**
Convert Primary to Standby	26 secs	21 sec
Convert Standby to Primary (C2P)	47 secs	7 secs
Open new Primary (OnP)	152 secs	14 secs
Open PDB and Start Service (OPDB)	130 secs	39 secs
Total App Downtime	355 secs or 5 minutes 55 secs	81 secs (78% drop)

For existing databases with a lot of data files, the following table compares the use of `ALTER TABLE MOVE` or `DBMS_REDEFINITION` to migrate tables or partitions to bigfile tablespaces.

Areas of Interest or Use Cases	**DBMS_REDEFINITION**	**ALTER TABLE MOVE ONLINE**
Application Impact	• No DDL changes allowed during move • Application blackout of seconds during activation	• No DDL changes allowed during move • Application blackout during final switch unknown

Areas of Interest or Use Cases	DBMS_REDEFINITION	ALTER TABLE MOVE ONLINE
Application Functionality Supported	• DML supported • PDML supported • No DDL changes allowed during move	• DML supported • PDML not supported • No DDL changes allowed during move
Impact of indexes	• Available during move • Indexes maintained	• Available during move • Indexes maintained after move (with `UPDATE INDEXES` clause) • Indexes moved separately (`REBUILD ONLINE`)
Space Requirements	Double space required (tables+indexes)	Double space required (tables+indexes)
Table Partition Functionality	Move entire partitioned table with one execution	Move partition by partition in order to maintain all indexes
Statistics Management	New statistics can be created before activation	New statistics created after activation
Monitoring Progress	You can query the `V$ONLINE_REDEF` view to monitor the progress of an online table redefinition operation.	Query `V$SESSION_LONGOPS?`
Resume on failure	Restart-able	Unknown
Rollback	Yes	N/A
Restrictions	• Tables with `LONG` columns can be redefined online, but those columns must be converted to `CLOBS`. Also, `LONG RAW` columns must be converted to `BLOBS`. Tables with `LOB` columns are acceptable. • Index-organized table can be moved • Domain indexes can be moved • Parallel DML and direct path `INSERT` operations are allowed Many 'corner case' restrictions with `DBMS_REDEFINITION`. See Restrictions for Online Redefinition of Tables in *Oracle Database Administrator's Guide*	• Cannot move a table with a `LONG` or `RAW` column • Cannot move partitioned index-organized table. • Cannot move if a domain index is defined on the table like spatial, XML, or a Text index. • Parallel DML and direct path `INSERT` operations are not possible during a table move. • Cannot move index-organized tables that contain any `LOB`, `VARRAY`, Oracle-supplied type, or user-defined object type columns.
Documentation and References	See DBMS_REDEFINITION in *Oracle Database PL/SQL Packages and Types Reference*	See ALTER TABLE in *Oracle Database SQL Language Reference*

Use Automatic Shared Memory Management and Avoid Memory Paging

Enable Automatic Shared Memory Management by setting the `SGA_TARGET` parameter, and set the `USE_LARGE_PAGES` database initialization parameter to `AUTO_ONLY` or `ONLY` and the `USE_LARGE_PAGES` ASM initialization parameter to `TRUE`.

Use the following guidelines in addition to setting `SGA_TARGET` to enable Automatic Shared Memory Management.

- The sum of SGA and PGA memory allocations on the database server should always be less than your system's physical memory while still accommodating memory required for processes, PGA, and other applications running on the same database server.
- To get an accurate understanding of memory use, monitor PGA memory and host-based memory use by querying `V$PGASTAT` for operating systems statistics.
- Avoid memory paging by adjusting the number of databases and applications, or reducing the allocated memory settings.

  Set `PGA_AGGREGATE_LIMIT` to specify a hard limit on PGA memory usage. If the `PGA_AGGREGATE_LIMIT` value is exceeded, Oracle Database first terminates session calls that are consuming the most untunable PGA memory. Then, if the total PGA memory usage is still over the limit, the sessions that are using the most untunable memory will be terminated.

Set the database initialization parameter `USE_LARGE_PAGES=AUTO_ONLY` or `ONLY`, and set the ASM initialization parameter `USE_LARGE_PAGES=TRUE`.

- Make sure that the entire SGA of a database instance is stored in HugePages by setting the `init.ora` parameter `USE_LARGE_PAGES=ONLY`, or set to `AUTO_ONLY` on Exadata systems.

  Setting `USE_LARGE_PAGES=ONLY` is recommended for database instances, because this parameter ensures that an instance will only start when it can get all of its memory for SGA from HugePages.
- For ASM instances leave the parameter `USE_LARGE_PAGES=ONLY` (the default value). This setting still ensures that HugePages are used when available, but also ensures that ASM as part of Grid Infrastructure starts when HugePages are not configured, or insufficiently configured.
- Use Automatic Shared Memory Management, because HugePages are not compatible with Automatic Memory Management.

Use Oracle Clusterware

Oracle Clusterware lets servers communicate with each other, so that they appear to function as a collective unit. Oracle Clusterware has high availability options for all Oracle databases including for single instance Oracle databases. Oracle Clusterware is one of minimum requirements in making applications highly available.

Oracle Clusterware provides the infrastructure necessary to run Oracle Real Application Clusters (Oracle RAC), Oracle RAC One Node, and Oracle Restart. Oracle Grid Infrastructure is the software that provides the infrastructure for an enterprise grid architecture. In a cluster, this software includes Oracle Clusterware and Oracle ASM.

For a standalone server, the Grid Infrastructure includes Oracle Restart and Oracle ASM. Oracle Restart provides managed startup and restart of a single-instance (non-clustered) Oracle database, Oracle ASM instance, service, listener, and any other process running on the server. If an interruption of a service occurs after a hardware or software failure, Oracle Restart automatically restarts the component.

Oracle Clusterware manages resources and resource groups to increase their availability, based on how you configure them. You can configure your resources and resource groups so that Oracle Clusterware:

- Starts resources and resource groups during cluster or server start
- Restarts resources and resource groups when failures occur
- Relocates resources and resource groups to other servers, if the servers are available

For more information, see *Oracle Clusterware Administration and Deployment Guide* topics, High Availability Options for Oracle Database and Making Applications Highly Available Using Oracle Clusterware.

10

Oracle Flashback Best Practices

Oracle Database Flashback Technologies is a unique and rich set of data recovery solutions that let the database reverse human errors by selectively and efficiently undoing the effects of a mistake.

Before Flashback was introduced to Oracle Database, it might have taken minutes to damage a database but hours to recover it. With Flashback, correcting an error takes about as long as it took to make it. In addition, the time required to recover from this error is not dependent on the database size, which is a capability unique to Oracle Database.

Flashback supports database recovery at all levels, including the row, transaction, table, and the entire database. Flashback provides an ever-growing set of features to view and rewind data back and forth in time, and address several critical high availability and disaster recovery use cases. The list of features and use cases, as well as some key examples, can be found in Oracle Flashback Technology.

The Flashback features give you the capability to query historical data, perform change analysis, and perform the self-service repair to recover from logical corruptions while the database is online. With Oracle Flashback Technology, you can indeed undo the past.

Oracle Flashback Performance Observations

After adopting the configuration and operational best practices and applying recommended patches, Oracle has observed the following performance observations when Flashback Database is enabled on the primary or standby databases.

- Flashing back a database or a PDB to the previous hour usually takes seconds and minutes, even with a very high workload. It finishes in a fraction of the time it takes to apply a given amount of redo. Here are some observations:
  - Flashing back a large batch workload consisting of 400 GB of changes completed in less than 5 minutes.
  - Flashing back of a heavy OLTP of 8GB of changes completed in less than 2 minutes.
  - Due to many variables, there is no rule-of-thumb or calculation to estimate the time to complete a flashback. The tests which produced these observations were done on Exadata to remove system bottlenecks such as storage I/O bandwidth.
- The impact on OLTP workload on the primary database is usually less than 5 percent.
- The impact of a large insert (batch inserts, for example) or direct load operations is usually less than 5 percent if Flashback block new optimization is in effect; otherwise, the impact can vary dramatically (2-40% impact), so testing is required.

  Refer to the Flashback use cases that mention block new optimization descriptions and exceptions in Oracle Flashback Performance Tuning for Specific Application Use Cases.
- Enabling Flashback database can reduce peak redo apply performance rates on a physical standby database if the standby system cannot handle the additional I/O throughput requirements in the Fast Recovery Area. However, even with Flashback database enabled on the standby, the achievable redo apply rates with Flashback enabled are still very high and can outperform application redo generation rates.

The following lists describe the critical flashback milestones and key performance improvements across different Oracle Database software releases:

Oracle Database 12c Release 2 (12.2)

- Flashback Pluggable Database enables the flashback of individual PDBs without affecting other PDBs.
- PDB Restore Points enable an ease of use method to set an alias to an SCN. This alias can then be used for flashback PDB or Point-In-Time Recovery.

Oracle Database 19c

- Creating a Restore Point on a primary database automatically propagates to a standby database, and creates a corresponding Restore Point on the standby database.
- When Flashback Database is enabled on both the primary and standby databases in an Oracle Data Guard configuration, flashing back the primary database causes the standby database to automatically flash back as well.

Oracle Database 21c

- Migrate Flashback Data Archive-enabled tables between different database releases
- Flashback Database support for data file resizing operations
- PDBs can be recovered to an orphan PDB incarnation within the same CDB incarnation or an ancestor incarnation

Oracle Flashback Configuration Best Practices

The following are Oracle MAA best practices for configuring Flashback technologies in Oracle Database.

Setting DB_FLASHBACK_RETENTION_TARGET

Set the `DB_FLASHBACK_RETENTION_TARGET` initialization parameter to the largest value prescribed by any of the following conditions that apply:

- To leverage Flashback database to reinstate your failed primary database after Oracle Data Guard failover, set `DB_FLASHBACK_RETENTION_TARGET` to a minimum of 60 (minutes) to enable reinstatement of a failed primary. When enabling Flashback database, a couple of hours are required to generate sufficient flashback data into the flashback logs before reinstatement is possible. You can query `V$FLASHBACK_DATABASE_LOG` to find the oldest flashback time.
- Consider cases where there are multiple outages (for example, a network outage, followed later by a primary database outage) that may result in a transport lag between the primary and standby databases at failover time. For such cases, set `DB_FLASHBACK_RETENTION_TARGET` to a value equal to the sum of 60 (mins) plus the maximum transport lag that you wish to accommodate. This ensures that the failed primary database can be flashed back to an SCN that precedes the SCN at which the standby became primary. This is a requirement for primary reinstatement.
- If you are using Flashback Database for fast point-in-time recovery from user error or logical corruptions, set `DB_FLASHBACK_RETENTION_TARGET` to a value equal to the farthest time in the past that you wish to be able to recover from.
- In most cases, `DB_FLASHBACK_RETENTION_TARGET` should be set to the same value on the primary and standby.

Sizing the Fast Recovery Area

Flashback Database uses its own logging mechanism, creating flashback logs and storing them in the Fast Recovery Area (FRA). Ensure that the FRA has allocated sufficient space to accommodate the flashback logs for the target retention size and for peak batch rates. Sizing the FRA is described in detail in the Oracle Backup and Recovery documentation, but generally the volume of flashback log generation is similar in magnitude to redo log generation. Use the following conservative formula and approach:

Target FRA = Current FRA + `DB_FLASHBACK_RETENTION_TARGET` x 60 x Peak Redo Rate (MB/sec)

Example:

- Current FRA or `DB_RECOVERY_FILE_DEST_SIZE`=1000GB
- Target `DB_FLASHBACK_RETENTION_TARGET`=360 (360 minutes)
- From AWR:
  - The peak redo rate for OLTP workload is 3 MB/sec for the database
  - The peak redo rate for the batch workload is 30 MB/sec for the database, and the longest duration is 4 hours
  - The worst-case redo generation size for a 6 hour window is (240 minutes x 30 MB/sec x 60 secs/min) + (120 minutes x 3 MB/sec x 60 secs/min) = 453,600 MB, or approximately 443 GB
- Proposed FRA or `DB_RECOVERY_FILE_DEST_SIZE`= 443 GB +1000 GB = 1443 GB

An additional method to determine FRA sizing is to enable Flashback Database and allow the database applications to run for a short period (2-3 hours), and then query `V$FLASHBACK_DATABASE_STAT.ESTIMATED_FLASHBACK_SIZE`.

Note that the `DB_FLASHBACK_RETENTION_TARGET` is a target, and there is no guarantee that you can flash back the database that far. The oldest flashback logs may be deleted if there is space pressure in the FRA where the flashback logs are stored. See Maintaining the Fast Recovery Area in *Oracle Database Backup and Recovery User's Guide* for a detailed explanation of the FRA deletion rules. You must use guaranteed restore points (GRP) to guarantee a flashback point-in-time. The required flashback logs will never be recycled or purged with GRP until GRP is dropped. The database can stop responding if there is a GRP but there is insufficient space, so you must allocate more space in the FRA depending on the intended duration of the GRP.

Configuring sufficient I/O bandwidth for Fast Recovery Area

Insufficient I/O bandwidth with Flashback Database on is usually indicated by a high occurrence of the "FLASHBACK BUF FREE BY RVWR" wait event in an Automatic Workload Repository (AWR) report for OLTP workloads and "FLASHBACK LOG FILE WRITE" latency > 30 ms for large insert operations.

In general, flashback I/Os are 1 MB in size. The overall write throughput would be similar to the redo generation rate if database force logging were enabled, or identical to your load rate for direct load operations. For simplicity, configure one large shared storage GRID and configure DATA on the outer portion of the disks or LUNS and RECO (FRA) on the inner amount of the disks or LUNS. This is done automatically for Exadata systems.

Setting LOG_BUFFER

To give Flashback Database more buffer space in memory, set the initialization parameter `LOG_BUFFER=256MB` or higher, depending on operating system limits.

Oracle Flashback Operational Best Practices

The following are Oracle MAA recommended operational best practices for Flashback Database.

- Gather database statistics using Automatic Workload Repository (AWR) or Oracle Enterprise Manager before and after enabling Flashback Database, so you can measure the impact of enabling Flashback Database.
- Using Oracle Enterprise Manager, set the Enterprise Manager monitoring metric,"Recovery Area Free Space (%)" for proactive alerts of space issues with the fast recovery area (FRA).
- To monitor the progress of a Flashback Database operation, you can query the `V$SESSION_LONGOPS` view. For example,

  ```
  select * from v$session_longops where opname like 'Flashback%';
  ```

  If more detail is required on the Flashback Database operation, generate a detailed trace of the Flashback Database operation in the DIAGNOSTIC_DEST trace directory for the database by setting database parameter `_FLASHBACK_VERBOSE_INFO=TRUE`.
- When using Flashback Database to perform repeated tests on a test database, it is recommended that you use Guaranteed Restore Points (GRP) only, without explicitly turning on Flashback Database. To minimize space usage and flashback performance overhead, follow this recommended approach:

  ```
  Create Guaranteed Restore Point (GRP)
  Execute test
  loop
       Flashback database to GRP
       Open resetlogs
       Create new GRP
       Drop old GRP
       Execute
  testEnd loop
  ```

- Follow the Oracle Data Guard redo apply best practices described in Redo Apply Troubleshooting and Tuning.

Oracle Flashback Performance Tuning for Specific Application Use Cases

Performance Tuning for OLTP Workloads

The "flashback buf free by RVWR" wait event only occurs when Flashback Database is enabled. A session waits for the recovery writer (RVWR) to write flashback data to the flashback logs on disk because the buffers are full. The session may need to wait until RVWR can free up the buffers.

Suppose this event becomes one of the top wait events for the database. In that case, it is typically because the file system or storage system for the Fast Recovery Area (FRA) has insufficient I/O bandwidth to accommodate additional I/O from the Flashback writes. Refer to the Flashback Database section in the *Oracle Database Backup and Recovery User's Guide* for tuning considerations and evaluate the corresponding I/O and storage statistics.

Table 10-1 Top 5 Timed Foreground Events

Event	Waits	Times	Average wait (ms)	% database time	Wait class
write complete waits	1,842	23,938	12995	33.68	Configuration
flashback buf free by RVWR	53,916	20,350	377	28.63	Configuration
cell single block physical read	3,029,626	16,348	5	23.00	User I/O
buffer busy waits	6,248	5,513	882	7.76	Concurrency
DB CPU		1,757		2.47	

Performance Tuning for Direct Path Operations

Look at the "flashback log write bytes" and "physical write bytes" system statistics found in `v$sysstat`, in your AWR reports, or Oracle Enterprise Manager.

- "flashback log write bytes" = The total size in bytes of Flashback Database data written by RVWR to Flashback Database logs
- "physical write bytes" = The total size in bytes of all disk writes from the database application activity (and not other kinds of instance activity).

If (flashback log write bytes) / (physical write bytes) < 5%, then Flashback is not impacting your performance.

Otherwise, evaluate any operational changes or bug fixes that will allow you to use the Flashback block new optimization feature (refer to performance observation section above). Furthermore, ignore the "flashback log file sync" wait event, even if it's one of the top wait events.

Example of block new optimization in effect

In this example:

- flashback log write bytes = 1,223,442,432
- physical write bytes = 184,412,282,880

The result of (flashback log write bytes) / (physical write bytes) = 0.0066 < 5%, implies that there's only a fraction of flashback data compared to the physical writes within this interval where there are direct load operations. Even in this case, the "flashback log file sync" wait event was the 2nd highest wait event in the database, as shown in the following table.

Table 10-2 Top 5 Timed Foreground Events

Event	Waits	Times	Average wait (ms)	% database time	Wait class
direct path write	136,553	7,875	58	39.12	User I/O
flashback log file sync	91,566	5,887	64	29.25	User I/O
DB CPU		3,092		15.36	
log buffer space	20,545	1,737	85	8.63	Configuration
gc buffer busy release	1,277	487	382	2.42	Cluster

Example of block new optimization not in effect

In this example:

- flashback log write bytes= 184,438,194,176
- physical write bytes =184,405,925,888

The result of (flashback log write bytes) / (physical write bytes) = 100% > 5%, implies that in this case all direct writes also result in flashback log writes. Listed here are the top wait events for this case.

Table 10-3 Top 5 Timed Foreground Events

Event	Waits	Times	Average wait (ms)	% database time	Wait class
flashback log file sync	170,088	22,385	132	52.04	User I/O
direct path write	278,185	8,284	30	19.26	User I/O
flashback buf free by RVWR	38,396	5,048	131	11.74	Configuration
direct path read	220,618	4,242	19	9.86	User I/O
DB CPU		2,788		6.48	

Performance Tuning for Conventional Load Operations

The following examples illustrate two conventional loads, one that uses block new optimization and one that does not.

Example of block new optimization not in effect

The example below does not use the block new optimization because of a truncate just before loading the table. The wait events for a conventional load without block new optimization show a fairly large amount of total wait time spent in "flashback log file sync". This is because of time needed to read them before the image of the block into the buffer cache and well as writing the block to the flashback log.

Table 10-4 Top 5 Timed Foreground Events

Event	Waits	Times	Average wait (ms)	% database time	Wait class
flashback log file sync	235,187	13,728	58	30.82	User I/O
direct path write	558,037	10,818	19	24.29	User I/O
direct path read	459,076	8,419	18	18.90	User I/O
DB CPU		6,171		13.85	
flashback buf free by RVWR	79,463	4,268	54	9.58	Configuration

Looking at the instance statistics below you can see very little increase in the statistics that track block new optimizations.

Statistic	Total	Per second	Per transaction
flashback cache read optimizations for block new	62	0.06	1.13
flashback direct read optimizations for block new	8	0.01	0.15
flashback log write bytes	177,533,280,256	177,245,433.67	3,227,877,822.84

Statistic	Total	Per second	Per transaction
flashback log writes	18,917	18.89	343.95

If the "flashback cache read optimizations for block new" is much smaller than "flashback log writes" then the block new optimization does not have an effect.

The best tuning recommendation for the above load operation would be to increase I/O bandwidth or, perhaps better, change the manner in which the load is performed so that it can take advantage of block new optimizations. You can also wait until you are outside the flashback retention target, or remove the object from recycle bin if it was dropped.

Example of block new optimization not effect

The wait events for a conventional load with block new optimization show a relatively small amount of total time spent in "flashback log file sync" compared to other database waits, as shown here.

Table 10-5 Top 5 Timed Foreground Events

Event	Waits	Times	Average wait (ms)	% database time	Wait class
direct path write	284,115	8,977	32	34.20	User I/O
DB CPU		6,284		23.94	
log buffer space	128,879	5,081	39	19.36	Configuration
flashback log file sync	139,546	3,178	23	12.11	User I/O
latch: redo allocation	95,887	1,511	16	5.76	Other

Looking at the instance statistics you can see that the statistics that track block new operations have significantly increased during the load.

Statistic	Total	Per second	Per transaction
flashback cache read optimizations for block new	329	0.53	9.68
flashback direct read optimizations for block new	698,410	1,116.43	20,541.47
flashback log write bytes	1,197,514,752	1,914,271.66	35,221,022.12
flashback log writes	18,951	30.29	557.38

11

Oracle Global Data Services Best Practices

Oracle Database Global Data Services (GDS) is a holistic automated workload management feature of Oracle Database.

GDS provides workload routing, load balancing, inter-database service failover, replication lag-based routing, role-based global services, and centralized workload management for a set of replicated databases that are globally distributed or located within the same data center.

You can use GDS to achieve these benefits without the need to integrate with multiple-point solutions or homegrown products. GDS provides optimized hardware and software utilization, better performance, scalability, and availability for application workloads running on replicated databases.

Introduction to Global Data Services

Numerous organizations have multiple copies of their production or standby databases either locally or in different geographical locations. This redundancy fulfills various business needs, such as ensuring uninterrupted availability, preparing for disaster recovery, localizing content and caching, scaling operations, optimizing performance for local users, and complying with local regulations. Oracle Active Data Guard and Oracle GoldenGate, which synchronize these copies, are Oracle's strategic disaster recovery and replication technologies that provide the lowest Recovery Time Objective (RTO) and Recovery Point Objectives (RPO).

Achieving high performance and high availability by distributing workload across multiple database replicas presents challenges that extend beyond the capabilities of the replication technology. The workload must be load-balanced to use resources effectively and achieve the best performance. Outages must be handled intelligently so that applications remain available and deliver the best possible quality of service should a replica go offline. In an ideal world, load balancing and high availability using a pool of replicated databases occurs seamlessly and optimally using real-time information available from the Oracle Database. This ideal is achievable using Oracle Global Data Services (GDS).

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

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

A GDS configuration contains multiple global service managers per region. The global service managers are “global listeners,” which understand real-time load characteristics and the user-defined service placement policies on the replicated databases. These global service

managers are instrumental in performing inter-database service failovers and load balancing of GDS. GDS is a shared infrastructure that can govern multiple sets of replicated databases. This documentation describes the configuration and operational practices for GDS. It is intended for enterprise architects, database architects, database administrators, technology managers, solution architects, application architects, and those who are influential in the overall database architecture design.

Global Data Services Concepts

Database services are logical abstractions for managing workloads in an Oracle database. Each service represents a workload with common attributes, service-level thresholds, and priorities. The grouping can be based on work characteristics, including the application function. For example, the Oracle E-Business Suite defines a service for each application module, such as general ledger, accounts receivable, and order entry. Services are built into Oracle Database, providing a single system image for workloads. Services enable administrators to configure a workload, administer it, enable and disable it, and measure the workload as a single entity. Clients connect using a database service name.

In Oracle RAC, a service can span one or more instances and facilitate workload balancing based on real-time transaction performance. This provides high availability, rolling changes by workload, and complete location transparency. For replicated environments, GDS introduces the concept of a "global service". Global services are provided across a set of databases containing replicated data that belongs to a particular administrative domain known as a GDS pool. Examples of a GDS pool are a SALES pool or an HR pool. A set of databases in a GDS configuration and the database clients are said to be in the same GDS region if they share the network proximity. Examples of GDS regions are the Asian region, European region, and so on.

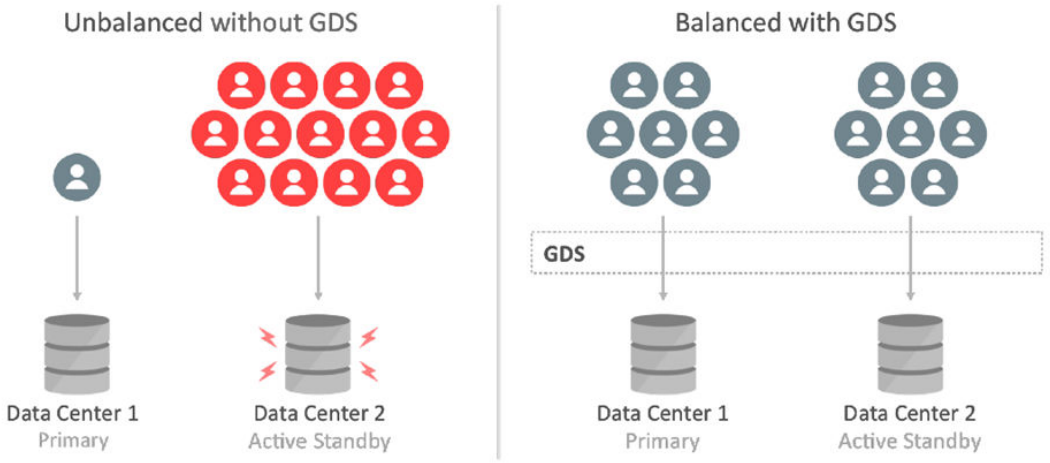
All of the characteristics of traditional database services are supported by global services. Global services extends traditional database services with additional attributes such as global service placement, replication lag (Oracle Active Data Guard and Oracle GoldenGate from 19c onwards), and region affinity.

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

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

Region affinity: A global service allows you to set preferences to which region (for example Asia or Europe) your given applications should connect.

Figure 11-1 Workload Balancing with Global Data Services



Key Capabilities of Global Data Services

Global Data Services (GDS) technology provides the following principal capabilities:

- Region-based workload routing: With GDS, you can choose to configure client connections to be routed among a set of replicated databases in a local region. This capability allows you to maximize application performance (avoiding the network latency overhead of accessing databases in remote areas).
- Connect-time load balancing: Global service managers use the load statistics from all databases in the GDS pool, inter-region network latency, and the configured connect-time load balancing goal to route the incoming connections to the best database in a GDS pool.
- Runtime load balancing: GDS enables runtime load balancing across replicated databases by publishing a real-time load balancing advisory for connection pool-based clients (for example, OCI, JDBC, ODP.NET, WebLogic, and so on.). The connection pool-based clients subscribe to this load-balancing advisory and route database requests in real time across already-established connections.

  With GDS's runtime connection load balancing feature, application client work requests are dynamically routed to the database that offers the best performance. In addition, GDS also supports the ability to dynamically re-distribute connections when the database performance changes.
- Inter-database service failover: If a database global service crash occurs, GDS, considering the service placement attributes, automatically performs an inter-database service failover to another available database in the pool. GDS sends Fast Application Notification (FAN) events so that the client connection pools can reconnect to the new database where the global service has been started.
- Replication lag-based workload routing: With Oracle Active Data Guard, standbys may lag behind the primary database. A global service allows you to choose the acceptable lag tolerance for a given application. GDS routes requests to a standby database whose lag is below the limit. If the lag exceeds the lag limit, the service is relocated to another available standby database that lags below the threshold. New requests are routed to a standby database that satisfies the lag limit. The global service is shut down if there is no available

database. When the lag is resolved or comes within the limit, GDS automatically brings up the service.

With Oracle GoldenGate replication, when the lag exceeds the lag threshold defined for a service (lag defined by `SELECT MAX(INCOMING_LAG FROM GGADMIN.GG_LAG)`, that service is stopped on that database. The service defines the effect of that; it may or may not terminate all the sessions based on the configuration. The service is restarted if the lag comes back within the threshold. After the service has been stopped, the global service manager automatically performs failover processing. Any new connections to this service are directed elsewhere than the lagged database. So, if there are two databases in the pool, and the service is `preferred_all` with `lag=10` initially, the service runs on both databases, and the connections are load-balanced. If the second database goes past the lag threshold, the service is stopped there, and any new connections are directed only to the first database. If the lag comes back within the threshold, the service is restarted, load balancing continues, and new connections can use the second database.

- Role-based global services: When a database role transition is performed with Oracle Data Guard Broker, GDS can automatically relocate the global service to the new primary database and the new standby if the role assigned to the service matches the role of the database.
- Centralized workload management for replicas: GDS allows more straightforward configuration and management of the replicated databases' resources located anywhere with a single unified framework.

Benefits of Global Data Services

Global Data Services (GDS) allows fault-tolerant database services to be deployed and centrally managed across a set of replicated databases. The GDS framework provides workload balancing across these databases. More specifically, GDS is an Oracle-integrated solution that renders the following benefits:

- Higher availability and global scalability support seamless inter-database service failover among replicated databases in any data center, yielding higher application availability.
- GDS provides application scalability on demand by allowing dynamic addition of databases. It enables replicated databases to be added to the GDS infrastructure dynamically and transparently to obtain additional resource capability to scale application workloads. GDS allows this with no change to the application configuration or client connectivity.

Better Performance and Elasticity

With integrated load balancing across multiple databases, GDS addresses inter-region resource fragmentation. Under-utilized resources in one region can be put to work on the workload of another region's over-utilized resources, achieving optimal resource utilization. GDS sends work requests to less powerful databases in a GDS pool containing replicated databases running on database servers of different processor generations and various resources (CPU, memory, I/O). When more powerful databases are overloaded, the goal should be to equalize the response time.

Improved Manageability

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

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

Application Workload Suitability for Global Data Services

Global Data Services (GDS) is best for replication-aware application workloads; it is designed to work in replicated environments. Applications that are suitable for GDS adoption possess any of the following characteristics:

- The application can separate its work into read-only, read-mostly, and read-write services to use Oracle Active Data Guard or Oracle GoldenGate replicas. GDS does not distinguish between read-only, read-write, and read-mostly transactions. The application connectivity has to be updated to separate read-only or read-mostly services from read-write services, and the GDS administrator can configure the global services on appropriate databases. For example, a reporting application can function directly with a read-only service at an Oracle Active Data Guard standby database.
- Administrators should be aware of and avoid or resolve multi-master update conflicts to take advantage of Oracle GoldenGate replicas. For example, an internet directory application with built-in conflict resolution enables the read-write workload to be distributed across multiple databases, each open read-write and synchronized using Oracle GoldenGate multi-master replication.
- Ideally, the application is tolerant of replication lag. For example, a web-based package tracking application that allows customers to track the status of their shipments using a read-only replica, where the replica does not lag the source transactional system by more than 10 seconds.

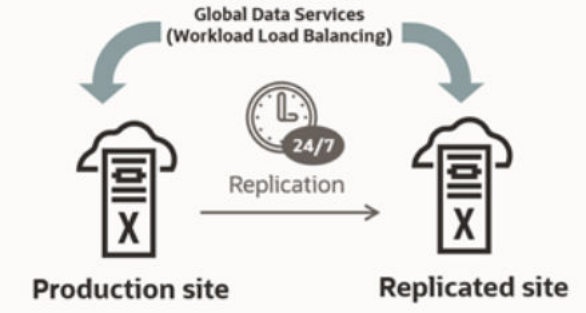
Global Data Services in Oracle Maximum Availability Architecture

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

Without Oracle Global Data Services, administrators were required to deploy 3rd party load balancers or have opted for custom-written connection managers. The 3rd party solutions come with significant integration costs, and the DIY homegrown solutions require high initial cost and time to value as well as an overhead of maintenance and support effort.

Using Global Data Services, you can unify replicated database resources within a single framework, avoiding the need to produce homegrown or 3rd party integration for load balancing. You can minimize the vendor integration touch points in their high availability and disaster recovery stack.

Figure 11-2 Oracle Global Data Services in Maximum Availability Architecture



Global Data Services is a strategic MAA component available within the Oracle Database. GDS is well integrated with the Oracle ecosystem, providing workload routing, load balancing, and service failover across replicated databases located within and across data centers. Simply put, GDS is a database load balancer for replicated databases and provides high availability through the inter-database service failover capability.

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

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

For detailed information about GDS, see the Global Data Services technical brief at http://oracle.com/goto/gds.

Partial or Full Site Outage with Global Data Services

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

A partial-site failure is when the application or database stack encounters an outage. Connecting clients to the disaster recovery site application stack can mitigate application stack failure. However, when the primary database becomes unavailable, the application tier connected to the primary database remains intact. All that is required to maintain availability is to redirect the application tier to the new primary database after a Data Guard failover has been completed. In this use case, the standby database is located within a distance from the surviving application tier such that it can deliver acceptable performance using a remote connection after a database failover has occurred.

Global Data Services (GDS) enables service management and load balancing between replicated databases *within* a region. However, the application tier still functions when GDS global service managers can route connections to the best available replica based on load balancing policies and service availability. By contrast, an out-of-region failover requires users to be directed to a remote application tier local to the new production database (serviced by a different set of in-region global service managers. This document focuses on GDS configuration for failover within a region.

Global Data Services Configuration

High-Level Deployment Steps

The following are the basic steps you would take to implement Global Data Services.

1. Install Global Data Services (GDS) global service manager software on global service manager servers.
   - Minimum of 1 global service manager for each region
   - Recommended 3 global service managers for each region
2. Pre-create the GDS catalog database.
3. Set up GDS Administrator accounts and privileges.
4. Configure GDS.
   - Create the GDS catalog and standby databases.
   - Add global service managers, regions, pools, databases, and global services.
5. Set up client connectivity.

Configuration Example

The following steps describe how to implement Global Data Services.

This example configuration of Global Data Services (GDS) uses an Administrator-managed Oracle RAC database. Administrator-managed deployment means that you configure database services to run on specific instances belonging to a particular database using a preferred and available designation.

Policy-managed deployment is based on server pools, where database services run within a server pool as singletons or uniformly across all of the servers in the server pool. Databases are deployed in one or more server pools, and the size of the server pools determines the number of database instances in the deployment. For detailed information about GDS, see Global Data Services Concepts and Administration Guide

1. Create and prepare a GDS catalog database.

   GDS uses a catalog database to store meta-data relating to the layout and status of the GDS configuration. For maximum availability, Oracle recommends that the GDS catalog database be deployed independently and that Oracle's high-availability features, such as Oracle Real Application Clusters (Oracle RAC) and Oracle Data Guard, be used to protect the catalog database against outages.
2. Create the `GSM_ADMIN` user and assign that user the `GSMADMIN_ROLE`.

Note that by default, the password for both GSM_ADMIN, GSMUSER, and GSMCATUSER expires after 180 days.

```
SQL> create user gsm_admin identified by password;

User created.

SQL> grant gsmadmin_role to gsm_admin;

Grant succeeded.

SQL> exit
```

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

```
GDSCAT =
(DESCRIPTION =
(ADDRESS = (PROTOCOL = TCP)(HOST = <hostmane>)(PORT = 1521))
(CONNECT_DATA =
(SERVER = DEDICATED)
(SERVICE_NAME = gdscat)
))
```

4. With the environment configured for the global service manager home, use GDSCTL to connect to and create the GDS catalog with Auto VNCR disabled (Auto VNCR can cause problems with Oracle RAC deployments).

```
GDSCTL>create catalog -database gdscat -user gsm_admin -autovncr OFF

"gsm_admin" password:

Catalog is created
```

5. Connect to the catalog database, unlock the GSMCATUSER user, and set the password.

```
SQL> alter user gsmcatuser account unlock;

User altered.

SQL> alter user gsmcatuser identified by password;

User altered.
```

6. With the environment configured for the global service manager home, use GDSCTL to connect to, create, and start the global service manager listeners.

   As a best practice, global service manager listeners should reside on hardware separate from that hosting the Oracle Databases in the GDS configuration. The resource requirements for hardware needed to run global service manager listeners are lightweight and can easily be accommodated using virtual machines.

```
GDSCTL>add gsm -gsm gsm1 -listener 1522 -catalog gdscat

"gsmcatuser" password:
```

```
Create credential oracle.security.client.connect_string1

GSM successfully added

GDSCTL>start gsm -gsm gsm1

GSM is started successfully

GDSCTL>status

Alias GSM1
Version 19.17.0.3.0
Start Date 13-APR-2023 09:40:59
Trace Level off
Listener Log File
 /u01/app/oracle/diag/gsm/hostname/gsm1/alert/log.xml
Listener Trace File
 /u01/app/oracle/diag/gsm/hostname/gsm1/trace/ora_64863_139739749930432.trc
Endpoint summary
(ADDRESS=(HOST=hostname.example.com)(PORT=1522)(PROTOCOL=tcp))
GSMOCI Version 0.6.11
Mastership Y
Connected to GDS catalog Y
Process Id 64883
Number of reconnections 0
Pending tasks. Total 0
Tasks in process. Total 0
Regional Mastership TRUE
Total messages published 0
Time Zone -04:00
Orphaned Buddy Regions: None
GDS region regionora
```

7. With the environment configured for the global service manager home, use GDSCTL to create a default GDS pool and default region.

```
GDSCTL>add gdspool -gdspool sales

GDSCTL>add region -region slc

GDSCTL>add region -region sca
```

8. With Auto VNCR disabled during GDS catalog creation to avoid issues, use GDSCTL to add hosts using the add invitednode command, using the host name or IP address appropriately.

```
GDSCTL>add invitednode 192.0.2.1

GDSCTL>add invitednode host1.example.com
```

9. Unlock the `GSMUSER` account.

Before adding a database to a pool, the database administrator should unlock the `GSMUSER` account and give the password to the GDS pool administrator, as shown in the following example.

```
SQL> alter user gsmuser account unlock;

User altered.

SQL> alter user gsmuser identified by password;

User altered.
```

10. Add databases to the GDS pool.

    To be part of a GDS pool, a database must use a server parameter file (SPFILE). An Oracle RAC database should also have SCAN set up.

    To add a database, connect to the GDS catalog using the GDS pool or GDS administrator credentials. For example, without Data Guard, the following add database command can be used.

```
GDSCTL>add database -connect mts -region sca -gdspool sales

Catalog connection is established

"gsmuser" password:

DB Unique Name: mts
```

> **Note:**
>
> When using Oracle Active Data Guard with GDS, use `add brokerconfig` instead of `add database`, and then use `modify database` to configure the standby database (see add brokerconfig). The syntax for these commands would be like the following.
>
> ```
> GDSCTL>add brokerconfig -connect <primary_db> -gdspool <dbpool> -
> region <dc> -pwd <gsmuser_pwd>
>
> GDSCTL>modify database -database <standby_db> -connect <dc> -
> gdspool <dbpool> -region <dc> -pwd <gsmuser_pwd>
> ```

Database instance registration with a global service manager succeeds only when the request originates from a valid node. If a host on which a database resides contains multiple network interfaces, the auto-configuration could register the wrong set of IP addresses, leading to the database registration being rejected.

11. Correct any rejected registration and properly discover all database instances.

If a firewall exists between the global service managers, and the databases and the ports are not opened, the registration fails. From the global service manageralert log, you will see entries similar to the following.

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

192.0.2.2

Listener(VNCR option 1) rejected Registration request from destination

192.0.2.3
```

You will find that the database object exists in the GDS catalog, but some or all instances associated with specific hosts are missing.

```
GDSCTL>databases

Database: "mts" Registered: Y State: Ok ONS: Y. Role: PRIMARY

Instances: 1 Region: slc

Registered instances:

sales%1
```

To correct the rejected registration and properly discover all database instances, run `add invitednode` using the rejected IP address listed in the global service manager alert log.

12. If there is a firewall between the global service managers and the database, then once the ports have been opened and verified using tnsping, issue the `add invitenode` command as shown here.

```
GDSCTL>add invitednode 192.0.2.3

GDSCTL>databases

Database: "mts" Registered: Y State: Ok ONS: Y. Role: PRIMARY

Instances: 2 Region: slc

Registered instances:

sales%1

sales%2
```

13. Create a service on the GDS pool databases.

    The GDSCTL `add service` command creates a service on the GDS pool databases.

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

If this is an Oracle RAC database being added with multiple instances, then you must modify the service to add the database instances.

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

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

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

14. Verify that the global service is running.

```
GDSCTL>services

Service "sales_sb.sales.oradbcloud" has 2 instance(s). Affinity: ANYWHERE

Instance "sales%1", name: "mts1", db: "mts", region: "slc", status: ready.

Instance "sales%2", name: "mts2", db: "mts", region: "slc", status: ready.
```

Configuration Best Practices

Oracle MAA recommends the following best practices for implementing Global Data Services:

- Each client communicates using an Oracle-integrated connection pool such as UCP, OCI, or ODP.NET. The connection pools will be notified about any service failovers and load balancing advisory notifications using Fast Application Notification Events.
- **Run three global service managers in each region**. Create three global service managers in each region so that if one global service manager goes down, you have two remaining global service managers to provide redundancy. Each global service manager should reside on separate hardware. Global service managers enable connection routing among replicated databases. A global service manager is a stateless, lightweight, and intelligent listener that can repopulate its metadata from the GDS catalog.
- **Protect the GDS catalog database with Oracle Data Guard.**The GDS catalog is a small (< 100 GBs) repository that hosts the metadata of the GDS configuration, regions, global service managers, global services, databases, and so on. MAA recommends that you set up a local Data Guard standby database configured with Maximum Availability database protection mode, Data Guard Fast-Start failover, and a remote physical standby database. All GDS catalog standby databases should use Oracle Active Data Guard for the best data protection and reside on separate hardware and storage.

Using FAN ONS with Global Data Services

Fast Application Notification (FAN) uses the Oracle Notification Service (ONS) for event propagation to all Oracle Database clients, including JDBC, Tuxedo, and listener clients.

ONS is installed as part of Oracle Global Data Services, Oracle Grid Infrastructure on a cluster, in an Oracle Data Guard installation, and when Oracle WebLogic is installed.

ONS propagates FAN events to all other ONS daemons it is registered with. No steps are needed to configure or enable FAN on the database server side, with one exception: OCI FAN and ODP FAN require that notification be set to TRUE for the service by GDSCTL. With FAN

auto-configuration at the client, ONS jar files must be on the CLASSPATH or in the ORACLE_HOME, depending on your client.

General Best Practices for Configuring FCF Clients

Follow these best practices before progressing to driver-specific instructions.

- Use a dynamic database service. Using FAN requires that the application connects to the database using a dynamic global database service. This is a service created using GDSCTL.
- Do not connect using the database service or PDB service. These services are for administration only and are not supported for FAN. The TNSnames entry or URL must use the service name syntax and follow best practices by specifying a dynamic database service name. Refer to the examples later in this document.
- Use the Oracle Notification Service when you use FAN with JDBC thin, Oracle Database OCI, or ODP.Net clients. FAN is received over ONS. Accordingly, in the Oracle Database, ONS FAN auto-configuration is introduced so that FCF clients can discover the server-side ONS networks and self-configure. FAN is automatically enabled when ONS libraries or jars are present.
- Enabling FAN on most FCF clients is still necessary in the Oracle Database. FAN auto-configuration removes the need to list the global service managers an FCF client needs.
- Listing server hosts is incompatible with location transparency and causes issues with updating clients when the server configuration changes. Clients already use a TNS address string or URL to locate the global service manager listeners.
- FAN auto-configuration uses the TNS addresses to locate the global service manager listeners and then asks each server database for the ONS server-side addresses. When there is more than one global service manager FAN auto-configuration contacts each and obtains an ONS configuration for each one.
- The ONS network is discovered from the URL when using the Oracle Database. An ONS node group is automatically obtained for each address list when `LOAD_BALANCE` is off across the address lists.
- By default, the FCF client maintains three hosts for redundancy in each node group in the ONS configuration.
- Each node group corresponds to each GDS data center. For example, if there is a primary database and several Oracle Data Guard standbys, there are by default three ONS connections maintained at each node group. The node groups are discovered when using FAN auto-configuration.

  With `node_groups` defined by FAN auto-configuration, and `load_balance=false` (the default), more ONS endpoints are not required. If you want to increase the number of endpoints, you can do this by increasing max connections. This applies to each node group. Increasing to 4 in this example maintains four ONS connections at each node. Increasing this value consumes more sockets.

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

- If the client is to connect to multiple clusters and receive FAN events from them, for example in Oracle RAC with a Data Guard event, then multiple ONS node groups are needed. FAN auto-configuration creates these node groups using the URL or TNS name. If automatic configuration of ONS (Auto-ONS) is not used, specify the node groups in the Oracle Grid Infrastructure or oraaccess.xml configuration files.

Client Side Configuration

As a best practice, multiple global service managers should be highly available. Clients should be configured for multiple connection endpoints where these endpoints are global service managers rather than local, remote, or single client access name (SCAN) listeners. For OCI / ODP .Net clients use the following TNS name structure.

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

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

Use the latest client driver aligned with the latest or older RDBMS for JDBC. Use one `DESCRIPTION` in the TNS names entry or URL Using more causes long delays connecting when `RETRY_COUNT` and `RETRY_DELAY` are used. Set `CONNECT_TIMEOUT=90` or higher to prevent logon storms for OCI and ODP clients.

Application-Level Configuration

Configuring FAN for Java Clients Using Universal Connection Pool

The best way to take advantage of FCF with the Oracle Database JDBC thin driver is to use the Universal Connection Pool (UCP) or WebLogic Server Active GridLink.

Setting the pool property `FastConnectionFailoverEnabled` on the Universal Connection Pool enables Fast Connection Failover (FCF). Active GridLink always has FCF enabled by default. Third-party application servers, including IBM WebSphere and Apache Tomcat, support UCP as a connection pool replacement.

For more information about embedding UCP with other web servers, see the following technical briefs.

- Design and deploy WebSphere applications for planned or unplanned database downtimes and runtime load balancing with UCP (https://www.oracle.com/docs/tech/database/planned-unplanned-rlb-ucp-websphere.pdf)
- Design and deploy Tomcat applications for planned or unplanned database downtimes and Runtime Load Balancing with UCP (https://www.oracle.com/docs/tech/database/planned-unplanned-rlb-ucp-tomcat.pdf)

Follow these configuration steps to enable Fast Connection Failover.

1. The connection URL must use the service name syntax and follow best practices by specifying a dynamic database service name and the JDBC URL structure (above and below).

   All other URL formats are not highly available. The URL may use JDBC thin or JDBC OCI.

2. If wallet authentication has not been established, remote ONS configuration is needed.

   Set the pool property `setONSConfiguration` in a property file as shown in the following example. The property file specified must contain an `ons.nodes` property and, optionally, properties for `oracle.ons.walletfile` and `oracle.ons.walletpassword`. An example of an `ons.properties` file is shown here.

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

3. Ensure the pool property `setFastConnectionFailoverEnabled=true` is set.
4. The `CLASSPATH` must contain ons.jar, ucp.jar, and the JDBC driver jar file.

   For example, ojdbc8.jar.
5. If you are using JDBC thin with Oracle Database, Application Continuity can be configured to failover the connections after FAN is received.
6. If the database needs different ONS endpoints than those autoconfigured, the ONS endpoints can be enabled.

   In a situation where multiple clusters exist with auto-ons enabled, auto-ons would generate node lists with the following guidelines:

   For EVERY active nodelist `oracle.ons.maxconnections` is set to 3 by default, so there is no need to set this explicitly. This example will result in the ONS client trying to maintain six total connections.

Configuring FAN for OCI Clients

OCI clients use ONS for FAN transport.

OCI clients embed FAN at the driver level so that all clients can use it regardless of the pooling solution. Ideally, both the server and the client would use release 19c or later.

Configuration for SQL*Plus and PHP

1. Set notification for the service.
2. For PHP clients only, add `oci8.events=On` to `php.ini`.

   Important: If xml is present with **events=-false** or events are not specified, this disables the usage of FAN. To maintain FAN with SQL*Plus and PHP when `oraccess.xml` is in use, set `events=-true`.
3. On the client side, using a client and Oracle Database, enable FAN in xml.

Configuration for OCI Clients

1. Tell OCI where to find ONS Listeners Starting.

The client installation comes with ONS linked into the client library. Using auto-config, the ONS endpoints are discovered from the TNS address. This automatic method is the recommended approach. Like ODP.Net, manual ONS configuration is also supported using `oraaccess.xml`.

2. Enable FAN high availability events for the OCI connections.

   To enable FAN you edit the OCI file xml to specify the global parameter events. This file is located in $ORACLE_HOME/network/admin. See Step 3: Ensure That FAN Is Used for more information.

3. Tell OCI where to find ONS Listeners.

   The client installation comes with ONS linked into the client library. Using auto-config, the ONS endpoints are discovered from the TNS address. This automatic method is the recommended approach. Like ODP.Net, manual ONS configuration is also supported using `oraaccess.xml`.

4. Enable FAN on the server for all OCI clients.

   It is still necessary to enable FAN on the database server for all OCI clients (including SQL*Plus).

Controlling Logon Storms

Small connection pools are strongly recommended, but controlling logon storms can be done with tuning when you have many connections.

Oracle MAA recommends the following tuning on servers that host Global Service Managers.

1. Increase the Listen backlog at the OS level.

   To have the new value take effect without rebooting the server, perform the following as root.

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

2. To persist the value across reboots, add this setting to /etc/sysctl.conf.

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

3. Increase `queuesize` for the global service manager listener.

   Update ora in Oracle home that the listeners are running from to increase the `queuesize` parameter:

   ```
   TCP.QUEUESIZE=6000
   ```

Graceful Application Switchover

Database services are used to manage workloads during the planned outage properly. Services must be properly created, and the application must obtain connections from a service.

These recommendations assume using a FAN-aware connection pool, such as Oracle Universal Connection Pool (UCP) to gracefully drain connections without application interruption from a service that is stopped. Your applications can use other connection types that don't support FAN-aware connection pools or have long-running transactions. Ideally, these applications will be disconnected before the maintenance window.

The recommendations below describe how to disconnect some sessions when their transaction ends in a timely manner or, ultimately, when the instance is shut down for maintenance.

The recommended and validated approach to understanding and optimizing your application's connection configuration is provided in the following sections; certain applications may have specific guidelines to follow.

Understanding Your Application's Use of Connections

Understanding how your application obtains and releases its connections is critical to determining whether it can gracefully switch to other instances in the cluster.

Find the following information about your application:

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

Services and Application Configuration Best Practices

You must have properly configured services and application attributes to perform a graceful switchover successfully. See My Oracle Support Doc ID 1617163.1 for a matrix of validated drivers and applications clients.

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

Using Oracle Active Data Guard with Global Data Services

Configure sessions to move in a rolling manner for Oracle Active Data Guard reader farm.

Prerequisites

You must have the following in place for this procedure to work correctly.

- Oracle Active Data Guard configuration using Oracle Database (release 19c or later recommended).
- Global Data Services (GDS) configuration using global service manager (release 19c or later recommended).
- A GDS service has been created to run on all Active Data Guard databases in the configuration.

  For example:

```
GDSCTL>add service -service sales_sb -preferred_all -gdspool sales
 -role physical_standby -notification TRUE
GDSCTL>modify service -gdspool sales -service sales_sb -database mts -
add_instances
 -preferred mts1,mts2
GDSCTL>modify service -gdspool sales -service sales_sb -database stm -
add_instances
 -preferred stm1,stm2
GDSCTL>start service -service sales_sb -gdspool sales
```

1. Check the current status of the services and related instances to ensure that services can be moved successfully.

   Note that the service should only be available on the source standby database at this point.

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

2. Stop services typically (not using the FORCE option) on the source database where connections are to be removed.
   - This step will quiesce the FAN-aware connection pools using FAN.
   - New connections are directed to other instances offering that service, and idle sessions are disconnected from the pool using the services.
   - Existing connections can continue until their work is complete and they are returned to the connection pool.

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

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

> **Note:**
>
> If you are performing a rolling upgrade of an Active Data Guard reader farm and the services are not running on other Active Data Guard reader nodes, you can complete the service stop on this database before performing the GDSCTL `stop service` described in this step.

3. Disconnect long-running sessions after the current query is completed.

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

4. Log on to the instance that you intend to shut down.

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

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

6. Run the `DBMS_SERVICE.DISCONNECT_SESSION` package for the service you stopped earlier.

   For example:

```
SQL> exec

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

7. Check `V$SESSION` again to ensure that sessions have logged off from the service.

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

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

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

9. Log on to the target database and check `V$SESSION` to see sessions connected to the service.

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

Using Oracle GoldenGate with Global Data Services

The following Oracle GoldenGate role transition example topology consists of two databases: GG replica1 and GG replica2. Oracle GoldenGate is set up with uni-directional replication, with Extract running initially on GG replica1 and Replicat running initially on GG replica2. The generic steps still apply for bi-directional GoldenGate replicas or downstream mining GoldenGate replicas.

Prerequisites

You must have the following in place for this procedure to work correctly.

- Oracle GoldenGate configuration that uses Oracle Database (19c or higher recommended)
- GoldenGate processes should not connect to the source or target database using the GDS service name, but a dedicated TNS alias. Using the GDS service will cause the database connections to terminate prematurely, causing possible data loss.
- A heartbeat table has been implemented in the GoldenGate source and target databases to track replication latency and ensure the Replicat applied SCN synchronization. The GoldenGate automatic heartbeat table feature should be enabled. Refer to the Oracle

GoldenGate Administration Guide for details on the automatic heartbeat table: https://docs.oracle.com/en/middleware/goldengate/core/19.1/gclir/add-heartbeattable.html.

- Global Data Services (GDS) configuration using global service manager (19c or higher recommended)
- GDS service has been created so that it can be run on all databases in the GoldenGate configuration.

  For example:

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

> **Note:**
>
> If you are using the lag tolerance option, specify the lag limit for the global service in seconds. Options for `add service` or `modify service` are `-lag {lag_value | ANY}`.

1. Check the current status of the services and related instances to ensure that they can be moved successfully.

   At this point, the service should only be available on the source database.

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

2. Stop services (not using the FORCE option) on the source database where connections are to be removed.
   - This step will quiesce the FAN-aware connection pools using FAN.
   - New connections are directed to other instances offering that service, and idle sessions are disconnected from the pool using the services.
   - Existing connections can continue until their work is complete and they are returned to the connection pool.

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

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

3. Disconnect long-running sessions after the current transaction is completed.

   Preferably, long-running batch jobs are scheduled to be stopped or queued before the maintenance window. This step handles long-running sessions that are still running and must be stopped abruptly (e.g., killed). Check with the application developers if these long-

running batch jobs are idempotent and recoverable before disconnecting long-running sessions.

4. Log on to the instance that you intend to shut down, and check `V$SESSION` to see if any sessions are still connected to the service.

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

5. Run the `DBMS_SERVICE.DISCONNECT_SESSION` package for the service you stopped earlier.

   For example:

```
SQL> exec

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

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

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

7. When all sessions associated with the GDS service have been disconnected, verify that all data from the GoldenGate source databases have been replicated to the target database.
   - Record the current database SCN from the GoldenGate SOURCE database.

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

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

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

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

The above steps allow for a graceful switchover. However, if this is a failover event where the source database is unavailable, you can estimate the data loss using the steps below.

1. When using the automatic heartbeat table, use the following query to determine the replication latency.

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

SOURCE       TARGET             GG Path
```

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

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

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

   Note that if the application workload can accept a certain level of data lag, it is possible to perform this step much earlier than step two listed above.

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

3. Log on to the target database and check `V$SESSION` to see sessions connected to the service.

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

Global Data Services Failover Across Regions Flow

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

- Flush the cache if the DNS server being used supports such a capability. The following is the flush capability of standard DNS BIND versions:
  - BIND 9.3.0: The command `rndc flushname name` flushes individual entries from the cache.
  - BIND 9.2.0 and 9.2.1: The cache can be flushed with the command `rndc flush`.
  - BIND 8 and BIND 9 up to 9.1.3: Restarting the named server clears the cache.

5. Refresh local DNS service caching.
   Some operating systems might cache DNS information locally in the local name service cache. If so, this cache must also be cleared to recognize DNS updates quickly.
6. The secondary site load balancer directs traffic to the secondary site middle-tier application server.

Global Data Services Limitations and Requirements

A Single GDS Manages	GDS Databases
• 5,000 GDS Pools • 10 GDS Regions • 5 global service managers per Region • 10,000 Database instances • 10,000 Global Services • 1,000 Mid-tier connection pools	• Can be a Single Instance or RAC • Can be CDB or Non-CDB • Can run on commodity or Engineered systems (Oracle Exadata, ODA) • Are managed with GDSCTL CLI or Enterprise Manager DB Plug-in • Must be licensed for Oracle Active Data Guard or Oracle GoldenGate

Requirement	Network Load Balancer	Oracle GDS
Locality-based routing	Yes	Yes
Connect-time database load balancing	Yes	Yes
Publish routing and failover intelligence to clients	No	Yes
Replication lag-based database workload routing	No	Yes
Inter-database global service failover	No	Yes
Automatic role-based global services	No	Yes
Centralized management of database services across replicas	No	Yes
Native integration for Oracle Active Data Guard	No	Yes
Cost effectiveness	Additional expenditure required	Included with Oracle Active Data Guard or Oracle GoldenGate license

Part III
Oracle RAC and Clusterware Best Practices

12

Overview of Oracle RAC and Clusterware Best Practices

Oracle Clusterware and Oracle Real Application Clusters (RAC) are Oracle's strategic high availability and resource management database framework in a cluster environment, and an integral part of the Oracle MAA Silver reference architecture.

Adding Oracle RAC to a Bronze MAA reference architecture elevates it to a Silver MAA reference architecture. The Silver MAA reference architecture is designed for databases that can’t afford to wait for a cold restart or a restore from backup, should there be an unrecoverable database instance or server failure.

The Silver reference architecture has the potential to provide zero downtime for node or instance failures, and zero downtime for most database and system software updates, that are not achievable with the Bronze architecture. To learn more about the Silver MAA reference architecture, see High Availability Reference Architectures.

Oracle Clusterware and Oracle RAC provide the following benefits:

- High availability framework and cluster management solution
  - Manages resources, such as Virtual Internet Protocol (VIP) addresses, databases, listeners, and services
  - Provides HA framework for Oracle database resources and non-Oracle database resources, such as third party agents
- Active-active clustering for scalability and availability
  - **High Availability** If a server or database instance fails, connections to surviving instances are not affected; connections to the failed instance quickly failover to surviving instances that are already running and open on other servers in the Oracle RAC cluster
  - **Scalability and Performance** Oracle RAC is ideal for high-volume applications or consolidated environments where scalability and the ability to dynamically add or re-prioritize capacity across more than a single server are required. An individual database may have instances running on one or more nodes of a cluster. Similarly, a database service may be available on one or more database instances. Additional nodes, database instances, and database services can be provisioned online. The ability to easily distribute workload across the cluster makes Oracle RAC the ideal complement for Oracle Multitenant when consolidating many databases.

The following table highlights various Oracle Clusterware and Real Application Cluster configuration best practices.

Table 12-1 Oracle RAC HA Use Cases and Best Practices

Use Case	Best Practices
Certified and validated Clusterware software stack	Use Oracle Clusterware and avoid third-party Clusterware. See *Oracle Database Clusterware Administration and Deployment Guide* Clusterware is built-in to all Oracle Exadata Systems.
Certified and validated storage architecture	Use Oracle Automatic Storage Management (Oracle ASM) and Oracle Advanced Cluster File System (Oracle ACFS) instead of third party volume managers and cluster file systems for the following MAA benefits: • Eliminate hot spots by distributing work across all disks • Scale and adjust storage capacity by adding and dropping disks and storage online • Reduce complexity by providing a simplified and uniform method (ASMCMD, ASMCA, ExaCLI, or oeadacli) to manage database storage • Inherent data corruption detection and repair when using ASM diskgroups • Simple management, patching and maintenance with an integrated Oracle Grid Infrastructure (Clusterware +ASM) without any additional drivers When using ASM with external redundancy, ensure that the underlying storage and network is highly available with no single point of failure. When using ASM native redundancy, high redundancy diskgroups are recommended to provide maximum protection for unplanned outages and during storage software updates. By default Exadata deployments use high redundancy for all diskgroups (both for data and recovery destinations). Oracle ACFS is a multi-platform, scalable file system and storage management technology that extends Oracle ASM functionality to support all customer files and can be leveraged for non-database files. These best practices are built-in to all Oracle Exadata Systems. See Introducing Oracle Automatic Storage Management

Table 12-1 (Cont.) Oracle RAC HA Use Cases and Best Practices

Use Case	Best Practices
Certified and validated network architecture	Ensure that the entire database and storage network topology has multiple network paths with no single point of failure. When connecting to the database service, use built-in Virtual Internet Protocol (VIP) addresses, Single Client Access Name (SCAN), and multiple local SCAN listeners configured over a bonded client network. Use a separate high bandwidth, bonded network for backup or Data Guard traffic. For the private network used as the cluster interconnect, Oracle recommends that non-Exadata customers use Oracle HAIP for network redundancy instead of using bonded networks. Bonding configurations have various attributes that behave differently with different network cards and switch settings. This recommendation does not apply to the private cluster interconnect in Exadata environments, because the bond setup has been properly configured and validated. Further, Exadata uses the `CLUSTER_INTERCONNECT` parameter over the highly available bonded network. Generic systems should NOT use the `CLUSTER_INTERCONNECT` and bonding but rather use Oracle HAIP.
Cluster configuration checks	Use Cluster Verification Utility (CVU) at monthly intervals to validate a range of cluster and Oracle RAC components such as shared storage devices, networking configurations, system requirements, and Oracle Clusterware. See Cluster Verification Utility Reference To perform a holistic, proactive health check and to evaluate if Oracle RAC or Exadata best practices are being followed, use `exachk` for Exadata RAC systems, or use `orachk` for non-Exadata RAC systems, at monthly intervals and before and after any software update. See ORAchk - Health Checks for the Oracle Stack (Doc ID 1268927.2)and Oracle Exadata Database Machine exachk or HealthCheck (Doc ID 1070954.1). Note that both `exachk` and `orachk` include CVU checks. `Exachk` covers software and configuration best practices and critical alerts for Storage, Network, Clusterware, ASM, and Database. Incorporate configuration recommendations from CVU, `exachk`, or `orachk`.
Reduce downtime for database node or instance failures	Typically, the default settings are sufficient for most use cases. If node detection and instance recovery need to be expedited, evaluate lower values for `FAST_START_MTTR_TARGET` Reducing `FAST_START_MTTR_TARGET` can increase database writer activity significantly, so additional I/O bandwidth is required. For Exadata systems, Instant Failure Detection capabilities use remote direct memory access (RDMA) to quickly confirm server failures in less than 2 seconds compared to typical 30 seconds detection found in most Oracle RAC clusters.

Table 12-1 (Cont.) Oracle RAC HA Use Cases and Best Practices

Use Case	Best Practices
Eliminate downtime for software updates	Use Oracle RAC rolling updates for Clusterware or database software updates (for example, Release Updates) to avoid downtime. Use out-of-place software updates when possible, so rollback and fallback use cases are simplified. Use software gold images to eliminate the complexity of running database `opatch` utility. For a fleet of databases on a single Oracle RAC cluster or multiple clusters, use Oracle Fleet Patching and Provisioning
Make application and processes highly available on the cluster	When an application, process, or server fails in a cluster, you want the disruption to be as short as possible and transparent to users. For example, when an application fails on a server, that application can be restarted on another server in the cluster, minimizing or negating any disruption in the use of that application. Similarly, if a server in a cluster fails, then all of the applications and processes running on that server must failover to another server to continue providing service to the users. Using the built-in `generic_application` resource type, Oracle Clusterware can manage all of these entities to ensure high availability, resource types or customizable scripts and application agent programs, and resource attributes that you assign to applications and processes. Use Oracle Clusterware to manage third-party resources and agents that reside on the cluster. See Making Applications Highly Available Using Oracle Clusterware
Reduce application downtime for planned and unplanned outages	Leverage Clusterware-managed services and application best practices to achieve zero application downtime. Use `SRVCTL` to manage services for your PDB. Never use default service for application connectivity. Always have at least one preferred Oracle RAC instance and at least one additional available Oracle RAC instance for High Availability. Applications should subscribe to HA Fast Application Notifications (FAN) and be configured to respond and failover if required. See Enabling Continuous Service for Applications and Continuous Availability - Application Checklist for Continuous Service for MAA Solutions
Capacity planning	Capacity planning and sizing should be done before deployment, and periodically afterward, to ensure that there are sufficient system resources to meet application performance requirements. Capacity planning needs to accommodate growth or consolidation of databases, additional application workloads, additional processes, or anything that strains existing system resources. Evaluating if performance requirements are still met during an unplanned outage or planned maintenance events is also crucial.

Part IV
Oracle Data Guard Best Practices

13

Overview of MAA Best Practices for Oracle Data Guard

By adding a physical standby database with Oracle Active Data Guard, a Silver MAA reference architecture is elevated to a Gold MAA reference architecture. Implement Oracle Data Guard best practices to achieve minimal downtime and potentially zero data loss for all unplanned outages.

Oracle Active Data Guard plays an important role in delivering the high availability and comprehensive data protection that you expect of the Gold MAA reference architecture. The Gold reference architecture, consisting of an Oracle RAC primary database and Oracle RAC standby systems with Oracle Active Data Guard, plus MAA configuration and life cycle operations, provides a comprehensive set of services that create, maintain, manage, and monitor one or more standby databases. Oracle Active Data Guard protects your data during all types of planned maintenance activities, such as software updates and major database upgrades, and unplanned outages, including database failures, site outages, natural disasters, and data corruptions.

The goal of Oracle Data Guard best practices is to help you implement tested and proven MAA best practices to ensure a successful and stable Data Guard deployment. The following steps connect you to the Oracle MAA best practices for planning, implementing, and maintaining this type of architecture.

1. Plan your Oracle Data Guard architecture, and take into account various considerations for the application, network, and so on.
2. Configure and Deploy Data Guard using Oracle MAA best practices.
3. Tune and Troubleshoot your Data Guard deployment.

To learn more about the Gold MAA reference architecture, see High Availability Reference Architectures.

14

Plan an Oracle Data Guard Deployment

Analyze your specific requirements, including both the technical and operational aspects of your IT systems and business processes, understand the availability impact for the Oracle Data Guard architecture options, and consider the impact of your application and network.

Oracle Data Guard Architectures

The Gold MAA reference architecture provides you with four architecture patterns, using Oracle Active Data Guard to eliminate single point of failure. The patterns vary from a single remote active standby with Fast Start Failover and HA Obeserver, to including far sync instances, multiple standbys, and reader farms.

When planning your Gold MAA Reference Architecture, see High Availability Reference Architectures for an overview of each Gold architecture pattern, and choose the elements to incorporate based on your requirements.

Application Considerations for Oracle Data Guard Deployments

As part of planning your Oracle Data Guard deployment, consider the resources required and application availability requirements in a fail over scenario.

Deciding Between Full Site Failover or Seamless Connection Failover

The first step is to evaluate which failover option best meets your business and application requirements when your primary database or primary site is inaccessible or lost due to a disaster.

The following table describes various conditions for each outage type and recommends a failover option in each scenario.

Table 14-1 Recommended Failover Options for Different Outage Scenarios

Outage Type	Condition	Recommended Failover Option
Primary Site Failure (including all application servers)	Primary site contains all existing application servers (or mid-tier servers) that were connected to the failed primary database.	Full site failover is required

Table 14-1 (Cont.) Recommended Failover Options for Different Outage Scenarios

Outage Type	Condition	Recommended Failover Option
Primary Site Failure (with some application servers surviving)	Some or all application servers are not impacted and the surviving application servers can reconnect to new primary database in a secondary disaster recovery site. Application performance and throughput is still acceptable with different network latency between application servers and new primary database in a secondary disaster recovery site. Typically analytical or reporting applications can tolerate higher network latency between client and database without any noticeable performance impact, while OLTP applications performance may suffer more significantly if there is an increase in network latency between the application server and database.	Seamless connection failover is recommended to minimize downtime and enable automatic application and database failover.
Complete Primary Database or Primary Server Failure	Application servers are not impacted and users can reconnect to new primary database in a secondary disaster recovery site. Application performance and throughput is still acceptable with different network latency between application servers and new primary database in a secondary disaster recovery site. Typically analytical or reporting applications can tolerate higher network latency between client and database without any noticeable performance impact, while OLTP applications performance may suffer more significantly if there is an increase in network latency between the application server and database.	If performance is acceptable, seamless connection failover is recommended to minimize downtime and enable automatic application and database failover. Otherwise, full site failover is required.

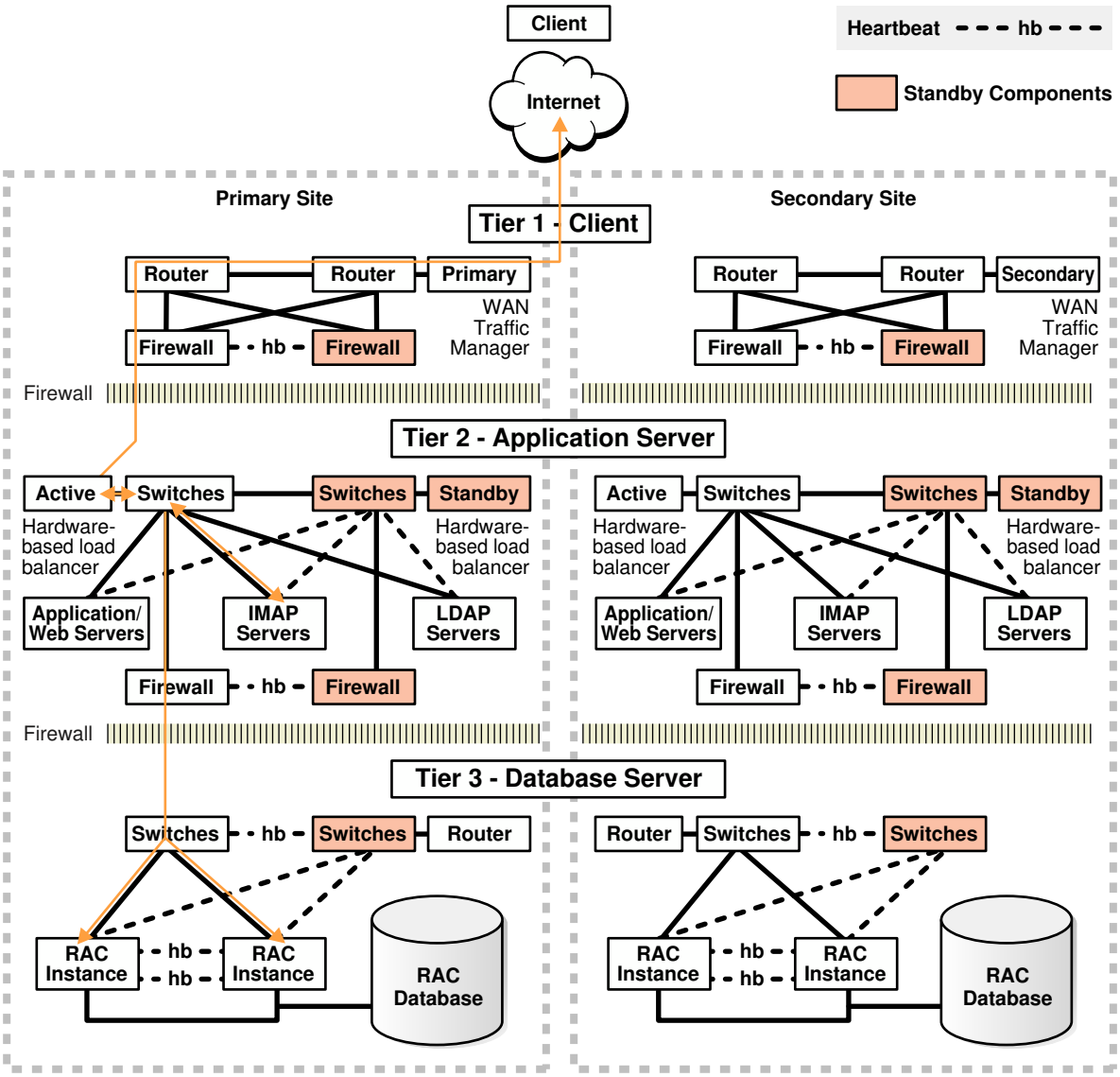
Full Site Failover Best Practices

A **full site failover** means that the complete site fails over to another site with a new set of application tiers and a new primary database.

Complete site failure results in both the application and database tiers becoming unavailable. To maintain availability, application users must be redirected to a secondary site that hosts a redundant application tier and a synchronized copy of the production database.

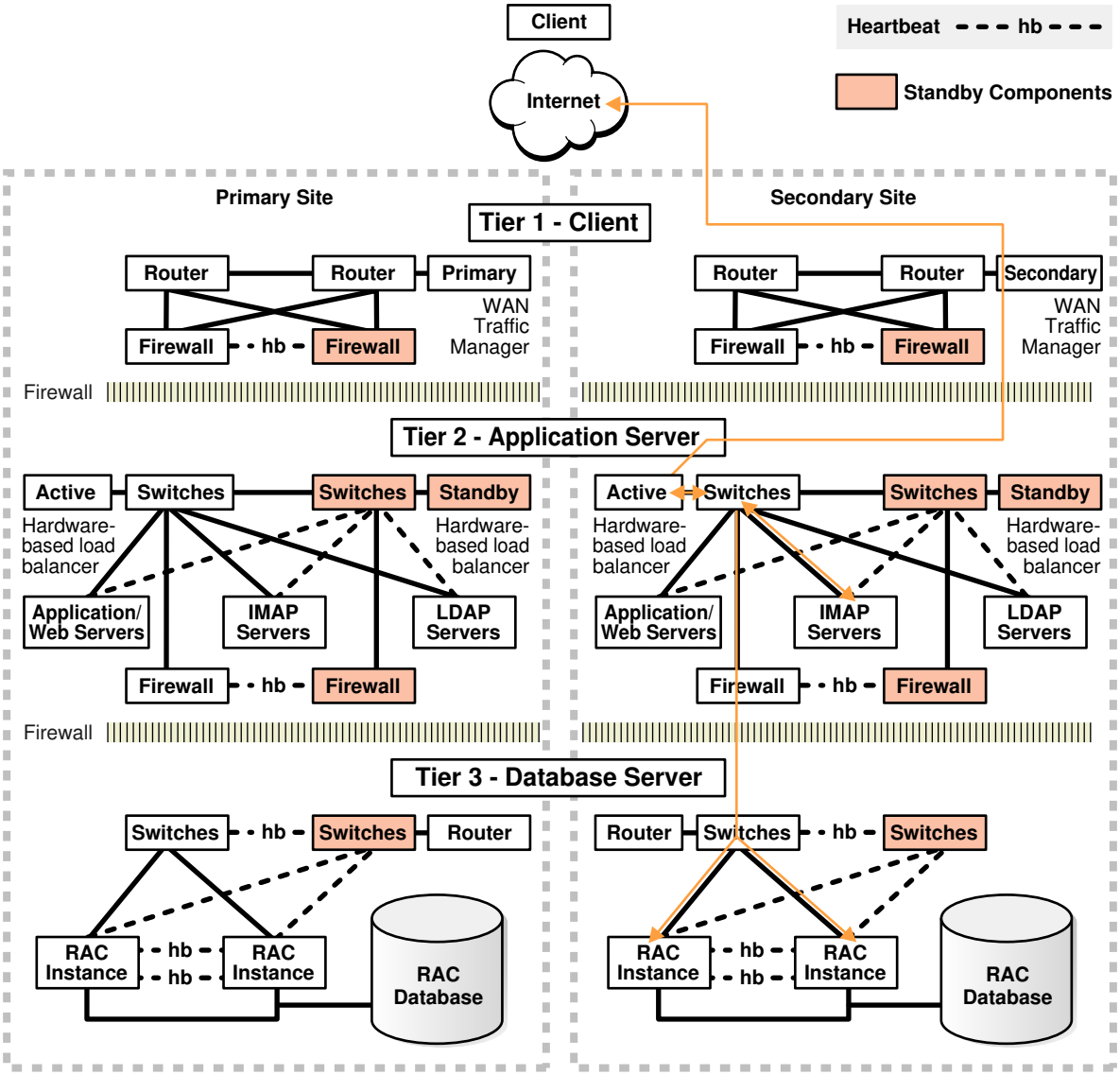
Consider the two figures below. The first figure shows the network routes before failover. Client or application requests enter the Primary site at the client tier, and are routed to the application server and database server tiers on the primary site.

Figure 14-1 Network Routes Before Site Failover



The second figure, below, illustrates the network routes after a complete site failover. Client or application requests enter the Secondary site at the client tier and follow the same path on the secondary site that they followed on the primary site.

Figure 14-2 Network Routes After Site Failover



MAA best practice is to maintain a running application tier at the standby site to avoid incurring start-up time, and to use Oracle Data Guard to maintain a synchronized copy of the production database. Upon site failure, a WAN traffic manager is used to perform a DNS failover (either manually or automatically) to redirect all users to the application tier at standby site while a Data Guard failover transitions the standby database to the primary production role.

Use Oracle Active Data Guard Fast-Start Failover to automate the database failover. Application server and non-database failovers can be automated and coordinated by using Oracle Site Guard. Oracle Site Guard orchestrates and automates any operations, such as starting up application servers on the secondary site, resynchronizing non-database meta data as Data Guard fails over automatically.

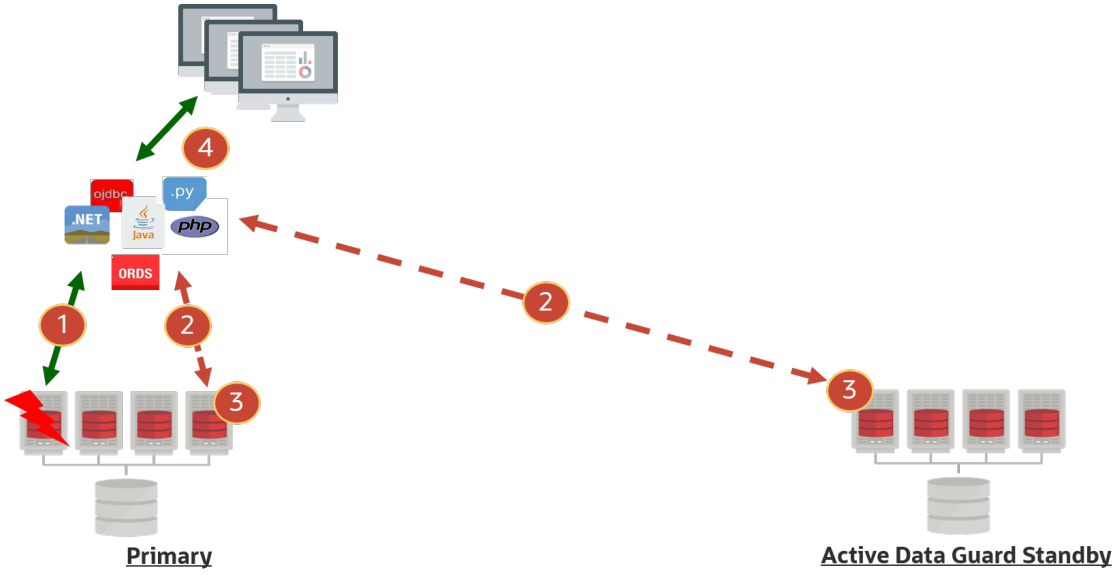
For more information about Oracle Site Guard, see the Oracle Site Guard Administrator's Guide.

Configuring Seamless Connection Failover

Automating seamless client failover in an Oracle Data Guard configuration includes relocating database services to the new primary database as part of a Data Guard failover, notifying clients that a failure has occurred to break them out of TCP timeout, and redirecting clients to the new primary database.

In the following figure, a database request is interrupted by an outage or timeout (1), so the session reconnects to the Oracle RAC cluster (2) (or standby) (2), the database request replays automatically on the alternate node (3), and the result from the database request is returned to the user (4).

Figure 14-3 Seamless Connection Failover



To achieve seamless connection failover, refer to Configuring Continuous Availability for Applications.

Assessing and Optimizing Network Performance

Oracle Data Guard relies on the underlying network to send redo from the primary database to standby databases. Ensuring that the network is healthy and capable of supporting peak redo generation rates helps avoid future transport lags.

A transport lag forms when the primary database cannot ship redo to the standby faster than primary instance's redo generation rate. A transport lag can lead to potential data loss if a primary database failure occurs.

Network assessment consists of evaluating

- Network reliability
- Network bandwidth to accommodate peak redo generation rates

> **Note:**
>
> Each instance of the primary database instance generates its own redo and ships redo to the standby database in a single network stream. Therefore, maximizing single process network throughput for each node is critical for redo transport.

Historically there are areas that can reduce network and redo transport throughput resulting in potential transport lags:

1. Network firewalls or network encryption

   Network firewalls and network (not Oracle Net) encryption can reduce overall throughput significantly. Verify throughput with the oratcp tool (described below), with and without encryption, and tune accordingly.

   At times reducing the encryption level can increase throughput significantly. A balance is required to meet security needs with your performance and data loss requirements.

2. Redo transport compression

   When database initialization parameter has `LOG_ARCHIVE_DEST_N` attribute `COMPRESSION=ENABLE`, Oracle background processes have to compress the redo before sending network message, and uncompress the redo before processing the redo. This reduces the overall redo and network throughput. Compression is only recommended if network bandwidth is insufficient between the primary and standby destinations.

3. Oracle Net encryption

   Depending on the Oracle Net encryption level, this will have varying redo throughput impact, because Oracle Net messages containing redo have to be encrypted before sending and then unencrypted before redo processing.

   Note that if database encryption is already enabled with Transparent Data Encryption (TDE), redo is already encrypted, although Oracle Net encryption can also encrypt the message headers.

4. Untuned network for redo transport

   - Increasing maximum operating system socket buffer size can increase single process throughput by 2-8 times. Test with different socket buffer sizes to see what value yields positive results, and ensure throughput is greater than the peak redo throughput.
   - Compare performance with various MTU settings.

     If average redo write size is less than 1500 bytes, then try various MTU settings including MTU=9000 (for example, Jumbo Frames) for network interface that sends or receives redo on your system. This may reduce some unnecessary network round trips which will increase overall throughput.

     Also note that for SYNC transport, Oracle's average redo write size (for example, Oracle message send) increases significantly as determined by v$sysstats or AWR statistics "redo size / redo writes".

     When sending redo across geographical regions, experiments have shown that using MTU=9000 can also benefit in some network topologies. Conduct performance tests with oratcp and compare the results with default MTU and MTU=9000 settings.

Gather Topology Information

Understanding the topology of the Oracle Data Guard configuration, and its relevance to Data Guard performance, helps eliminate infrastructure weaknesses that are often incorrectly attributed to the Data Guard architecture.

Oracle recommends that you outline the following high-level architecture information.

- Describe the primary and standby database system (number of nodes in Oracle RAC cluster, CPUs and memory per database node, storage I/O system)
- Describe network topology connecting the primary and standby systems
  - Network switches and firewalls in between primary and standby
  - Network bandwidth and latency

For standby databases with symmetric hardware and configuration, and with a well tuned network configuration that can support peak redo generation rates, the transport lag should be less than 1 second.

Understanding Network Usage of Data Guard

The phases of the Data Guard life cycle which use the network most heavily are:

- Instantiation - During this phase of standby database creation, files can be copied using parallelism from any host. Determining the degree of parallelism which maximizes throughput between nodes helps to optimize the standby instantiation process.
- Redo Transport (Steady State)- Under normal Data Guard operation the primary database ships redo to the standby which is then applied. Each RAC instance of a primary database ships redo in a single stream from the host on which it is running. Understanding the requirements of each primary database instance and ensuring a single process can achieve the throughput requirements is critical to a standby database staying current with the primary database.

Understanding Targets and Goals for Instantiation

Instantiation for large databases can take hours, or in extreme cases days. To allow for planning of an instantiation and also maximize throughput between the primary and standby system to complete the instantiation is as timely a manner as possible, first determine the goal for instantiation time. Then follow the process defined below to maximize per process throughput and identify the optimal degree of parallelism between the primary and standby nodes.

Understanding Throughput Requirements and Average Redo Write Size for Redo Transport

Required network bandwidth of a given Data Guard configuration is determined by the redo generate rate of the primary database.

> **Note:**
>
> In cases where the primary database is pre-existing, a baseline for the required network bandwidth can be established. If there is no existing primary database, skip this step and future references to the data further in the process.

While the Automatic Workload Repository (AWR) tool can be used to determine the redo generation rate, the snapshots are often 30 or 60 minutes apart which can dilute the peak rate. Since peak rates often occur for shorter periods of time, it is more accurate to use the following query which calculates the redo generation rate for each log when run on an existing database. (change the timestamps as appropriate)

```
SQL> SELECT THREAD#, SEQUENCE#, BLOCKS*BLOCK_SIZE/1024/1024 MB,
(NEXT_TIME-FIRST_TIME)*86400 SEC,
 (BLOCKS*BLOCK_SIZE/1024/1024)/((NEXT_TIME-FIRST_TIME)*86400) "MB/S"
FROM V$ARCHIVED_LOG
WHERE ((NEXT_TIME-FIRST_TIME)*86400<>0)
AND FIRST_TIME BETWEEN TO_DATE('2022/01/15 08:00:00','YYYY/MM/DD HH24:MI:SS')
 AND TO_DATE('2022/01/15 11:00:00','YYYY/MM/DD HH24:MI:SS')
AND DEST_ID=1 ORDER BY FIRST_TIME;
```

Example output:

```
THREAD# SEQUENCE# MB          SEC         MB/s
------- --------- ---------- ---------- ----------
      2      2291 29366.1963        831  35.338383
      1      2565 29365.6553        781 37.6000708
      2      2292 29359.3403        537  54.672887
      1      2566 29407.8296        813 36.1719921
      2      2293 29389.7012        678 43.3476418
      2      2294 29325.2217       1236 23.7259075
      1      2567 11407.3379       2658 4.29169973
      2      2295 24682.4648        477 51.7452093
      2      2296 29359.4458        954 30.7751004
      2      2297 29311.3638        586 50.0193921
      1      2568 3867.44092       5510 .701894903
```

> **Note:**
>
> To find the peak redo rate, choose times during the highest level of processing, such as peak OLTP periods, End of Quarter batch processing or End of Year batch processing.

In this short example the highest rate was about 52MB/s. Ideally the network will support the maximum rate plus 30% or 68MB/s for this application.

Verify Average Redo Write Size

Using `v$sysstats` or looking at your AWR reports for various workload and peak intervals, record the average redo write size based on

Average Redo Write Size = "REDO SIZE" / "REDO WRITES"

Use this average redo write size in your `oratcp` experiments. If the average redo write size > 1500 bytes, experiment with various MTU settings.

Understand Current Network Throughput

The Oracle utility oratcptest is a general-purpose tool for measuring network bandwidth and latency similar to iperf/qperf which can be run by any OS user.

The oratcptest utility provides options for controlling the network load such as:

- Network message size
- Delay time between messages
- Parallel streams
- Whether or not the oratcptest server should write messages on disk.
- Simulating Data Guard SYNC transport by waiting for acknowledgment (ACK) of a packet or ASYNC transport by not waiting for the ACK.

> **Note:**
>
> This tool, like any Oracle network streaming transport, can simulate efficient network packet transfers from the source host to target host similar to Data Guard transport. Throughput can saturate the available network bandwidth between source and target servers. Therefore, Oracle recommends that short duration tests are performed and that consideration is given for any other critical applications sharing the same network.

Measure the Existing Throughput of One and Many Processes

Do the following tasks to measure the existing throughput.

Task 1: Install oratcptest

1. Download the oratcptest.jar file from MOS note 2064368.1
2. Copy the JAR file onto both client (primary) and server (standby)

> **Note:**
>
> oratcptest requires JRE 6 or later

3. Verify that the host has JRE 6 or later
4. On all primary and standby hosts, verify that the JVM can run the JAR file by displaying the help

```
# java -jar oratcptest.jar -help
```

Task 2: Determine the Existing Throughput for a Single Process

Data Guard asynchronous redo transport (ASYNC) uses a streaming protocol which does not wait for packet acknowledgment and therefore achieves higher rates than SYNC transport.

1. Start the test server on the receiving (standby) side.

```
java -jar oratcptest.jar -server [IP of standby host or VIP in RAC
 configurations] -port=<any available port number>
```

2. Run the test client. (Change the server address and port number to match that of your server started in step 4.)

```
$ java -jar oratcptest.jar [IP of standby host or VIP in RAC
configurations]
 -port=<port number> -mode=async -duration=120 -interval=20s

[Requesting a test]
        Message payload        = 1 Mbyte
        Payload content type   = RANDOM
        Delay between messages = NO
        Number of connections  = 1
        Socket send buffer     = (system default)
        Transport mode         = ASYNC
        Disk write             = NO
        Statistics interval    = 20 seconds
        Test duration          = 2 minutes
        Test frequency         = NO
        Network Timeout        = NO
        (1 Mbyte = 1024x1024 bytes)

(17:54:44) The server is ready.
                    Throughput
(17:55:04)     20.919 Mbytes/s
(17:55:24)     12.883 Mbytes/s
(17:55:44)     10.457 Mbytes/s
(17:56:04)     10.408 Mbytes/s
(17:56:24)     12.423 Mbytes/s
(17:56:44)     13.701 Mbytes/s
(17:56:44) Test finished.
               Socket send buffer = 2 Mbytes
                  Avg. throughput = 13.466 Mbytes/s
```

In this example the average throughput between these two nodes was about 13 MB/s which does not meet the requirements of 68 MB/s from the query.

> **Note:**
>
> This process can be scheduled to run at a given frequency using the `-freq` option to determine if the bandwidth varies at different times of the day. For instance setting `-freq=1h/24h` will repeat the test every hour for 24 hours.

Task 3: Determine Existing Throughput for Multiple Processes

1. Repeat the previous test with two (2) connections (using num_conn parameter).

```
$ java -jar oratcptest.jar <target IP address> -port=<port number>
 -duration=60s -interval=10s -mode=async [-output=<results file>] -
num_conn=2
```

```
[Requesting a test]
        Message payload        = 1 Mbyte
        Payload content type   = RANDOM
        Delay between messages = NO
        Number of connections  = 2
        Socket send buffer     = (system default)
        Transport mode         = ASYNC
        Disk write             = NO
        Statistics interval    = 20 seconds
        Test duration          = 2 minutes
        Test frequency         = NO
        Network Timeout        = NO
        (1 Mbyte = 1024x1024 bytes)

(18:08:02) The server is ready.
                    Throughput
(18:08:22)     44.894 Mbytes/s
(18:08:42)     23.949 Mbytes/s
(18:09:02)     25.206 Mbytes/s
(18:09:22)     23.051 Mbytes/s
(18:09:42)     24.978 Mbytes/s
(18:10:02)     22.647 Mbytes/s
(18:10:02) Test finished.
          Avg. socket send buffer = 2097152
        Avg. aggregate throughput = 27.454 Mbytes/s
```

2. Re-run step 1 Iteratively and increase the value of num_conn by two each time until the aggregate throughput does not increase for three consecutive values. For example if the aggregate throughput is approximately the same for 10, 12 and 14 connections, stop.

> **Note:**
>
> RMAN can utilize all nodes in the cluster for instantiation. To find the total aggregate throughput, see My Oracle Support Creating a Physical Standby database using RMAN restore database from service (Doc ID 2283978.1).

3. Run the same test with all nodes in all clusters to find the current total aggregate throughput. Node 1 of primary to node 1 of standby, node 2 to node 2, etc. Sum the throughput found for all nodes.
4. Reverse the roles and repeat the tests.
5. Note the number of connections which achieved the best aggregate throughput.

Use the total size of the database and total aggregate throughput to estimate the amount of time it will take to complete the copy of the database. A full instantiation also needs to apply the redo generated during the copy. Some additional percentage (0%-50%) should be added to this estimated time based on how active the database is.

If the estimated time meets the goal, no additional tuning is required for instantiation.

Optimizing Redo Transport with One and Many Processes

If throughput from the prior single and multiple process tests meet the targets, no additional tuning is required. If higher throughput is required, setting the maximum TCP socket buffers size to a larger value is the primary method to potentially increase throughput.

Setting TCP Socket Buffer Size

The TCP socket buffers are system memory buffers which temporarily store incoming and outgoing data. Outgoing data is stored on the write buffers while incoming data is stored on the read buffers. Read and write socket buffers are allocated separately. When a buffer, generally the read buffer, fills up (often do to the application not pulling data out of the buffer fast enough), a message is sent to the sender to slow down or stop sending data. Allocating a larger buffer often improves redo transport by giving the application time to pull data off the wire without stopping the sender.

Tuning TCP socket buffer size is the primary approach to improving ASYNC transport and can improve SYNC transport as well in some cases.

> **Note:**
>
> With larger socket buffer sizes, TCP selective acknowledgment (SACK) is strongly recommended. Often times this is enabled by default but refer to your operating system documentation for details on confirming or enabling TCP selective acknowledgment.

To set TCP Socket Buffer Size do the following tasks.

Task 1: Determine Optimal Maximum Socket Buffer Size

Find the optimal maximum socket buffer size for a single process on the target network link by running a series of tests.

> **Note:**
>
> Bandwidth Delay Product is the product of the network link capacity of a channel and the round time, or latency. The minimum recommended value for socket buffer sizes is 3*BDP, especially for a high-latency, high-bandwidth network. Use oratcptest to tune the socket buffer sizes.

Task 2: Set Maximum Socket Buffer Size Temporarily

On the primary and standby systems follow these steps to set the maximum socket buffer size for requests. This will be done in memory and will not persist if the server is restarted for any reason.

Do the following steps as root.

1. First find the current size of the kernel parameters net.ipv4.tcp_rmem and net.ipv4.tcp_wmem. The values returned are the minimum, default and maximum size for socket buffers which TCP dynamically allocates. If a process requires more than the

default given when a socket is created, more buffers will be dynamically allocated up to the maximum value.

```
# cat /proc/sys/net/ipv4/tcp_rmem
4096    87380   6291456

# cat /proc/sys/net/ipv4/tcp_wmem
4096    16384   4194304
```

2. Change the values to 16MB or whatever 3*BDP was calculated to be

```
# sysctl -w net.ipv4.tcp_rmem='4096 87380 16777216';

# sysctl -w net.ipv4.tcp_wmem='4096 16384 16777216';
```

> **Note:**
>
> Increasing these values can increase system memory usage of any network socket on the system.

> **Note:**
>
> Changes made with sysctl are not permanent. Update the /etc/sysctl.conf file to persist these changes through machine restarts. There will be a step to change the configuration file at the end of this process once the proper setting is determined.

Task 3: Test Throughput of a Single Process

Re-run the previous tests allowing the socket buffers to dynamically grow to the new maximum set in the previous step

(as oracle)

Server (standby):

```
$ java -jar oratcptest.jar -server [IP of standby host or VIP in RAC
configurations]
 -port=<port number>
```

Client (primary):

```
$ java -jar oratcptest.jar <IP of standby host or VIP in RAC configurations>
 -port=<port number> -mode=async -duration=120s -interval=20s
```

> **Note:**
>
> Do not use the oratcptest `sockbuf` parameter because the kernel parameters which govern explicit requests for socket buffer size are different than those set for this test.

After the test completes the results from the client and server show the value for socket buffers during that test. At the time of this writing, that value is half of the actual socket buffer size and should be doubled to find the actual size used.

Client

```
[Requesting a test]
        Message payload = 1 Mbyte
        Payload content type = RANDOM
        Delay between messages = NO
        Number of connections = 1
        Socket send buffer = 2 Mbytes
        Transport mode = ASYNC
        Disk write = NO
        Statistics interval = 20 seconds
        Test duration = 2 minutes
        Test frequency = NO
        Network Timeout = NO
        (1 Mbyte = 1024x1024 bytes)
(11:39:16) The server is ready.
                Throughput
(11:39:36) 71.322 Mbytes/s
(11:39:56) 71.376 Mbytes/s
(11:40:16) 72.104 Mbytes/s
(11:40:36) 79.332 Mbytes/s
(11:40:56) 76.426 Mbytes/s
(11:41:16) 68.713 Mbytes/s
(11:41:16) Test finished.

          Socket send buffer = 8388608
            Avg. throughput = 73.209 Mbytes/s
```

Server

```
The test terminated. The socket receive buffer was 8 Mbytes.
```

> **Note:**
>
> oratcptest is reporting half of the buffers allocated to the socket. Double the number reported for the actual socket buffer size used during the test.

Task 4: Test Throughput of Multiple Processes

Now repeat the test using the num_conn value determined in the first tests. for example,. if the peak aggregate throughput was reached with 10 processes set num_conn=10.

Client

```
$ java -jar oratcptest.jar <IP of standby host or VIP in RAC configurations>
 -port=<port number> -mode=async -duration=120s -interval=20s -num_conn=10

[Requesting a test]
        Message payload        = 1 Mbyte
        Payload content type   = RANDOM
```

```
        Delay between messages = NO
        Number of connections  = 10
        Socket send buffer     = (system default)
        Transport mode         = ASYNC
        Disk write             = NO
        Statistics interval    = 20 seconds
        Test duration          = 2 minutes
        Test frequency         = NO
        Network Timeout        = NO
        (1 Mbyte = 1024x1024 bytes)

(19:01:38) The server is ready.
                   Throughput
(19:01:58)    266.077 Mbytes/s
(19:02:18)    242.035 Mbytes/s
(19:02:38)    179.574 Mbytes/s
(19:02:58)    189.578 Mbytes/s
(19:03:18)    218.856 Mbytes/s
(19:03:38)    209.130 Mbytes/s
(19:03:38) Test finished.
          Avg. socket send buffer = 8 Mbytes
        Avg. aggregate throughput = 217.537 Mbytes/s
```

> **Note:**
>
> oratcptest is reporting half of the buffers allocated to the socket. Double the number reported for the actual socket buffer size used during the test.

Server (Each connection will have the receive buffer printed. Double the socket buffer size in each instance)

```
The test terminated. The socket receive buffer was 8 Mbytes.

The test terminated. The socket receive buffer was 8 Mbytes.

The test terminated. The socket receive buffer was 8 Mbytes.

The test terminated. The socket receive buffer was 8 Mbytes.

The test terminated. The socket receive buffer was 8 Mbytes.

The test terminated. The socket receive buffer was 8 Mbytes.

The test terminated. The socket receive buffer was 8 Mbytes.

The test terminated. The socket receive buffer was 8 Mbytes.

The test terminated. The socket receive buffer was 8 Mbytes.

The test terminated. The socket receive buffer was 8 Mbytes.
```

Use the total size of the database and total aggregate throughput to estimate the amount of time it will take to complete the copy of the database. A full instantiation also needs to apply

the redo generated during the copy. Some additional percentage (0%-50%) should be added to this estimated time based on how active the database is.

Task 5: Repeat the Tests

Repeat the previous two tests with higher values for tcp_rmem and tcp_wmem if more throughput is needed. Understand that these higher values are available for other sockets as well but will be dynamically allocated only if needed. The table shows sample data tracking the different throughput results for different socket buffer sizes.

tcp_rmem maximum	tcp_wmem maximum	Single Process Throughput	Single Node Multi-Process Maximum Aggregate Throughput	Single Node Multi-Process Parallelism
6291456	4194304	13.5 MB/s	203 MB/s	16
8388608	8388608	48 MB/s	523 MB/s	14
16777216	16777216	73 MB/s	700 MB/s	14
33554432	33554432	132 MB/s	823 MB/s	14

Task 6: Set Parameters Permanently

Changes using sysctl modify the values in memory which do not persist through a reboot of the host. Once the optimal size for socket buffers is determined, set the kernel parameters so they persist through server restarts by editing the /etc/sysctl.conf file.

This must be done on all nodes of the primary and standby systems.

To make these changes persistent, edit the /etc/sysctl.conf either modifying the existing values or adding these values to the file if they are absent.

```
net.ipv4.tcp_rmem='4096 87380 16777216'

net.ipv4.tcp_wmem='4096 16384 16777216'
```

Task 7: Evaluate Larger MTU

Determine the network interfaces that are used by the Data Guard transport.

If Average Redo Write Size > current MTU setting (for example, typically the default 1500), evaluate if jumbo frames (for example, MTU=9000) can reduce the network RTT for these large network packets and improve overall redo throughput.

Shown here is an example of changing the MTU for Data Guard transport network interface for testing purposes on Linux.

```
ifconfig bondeth0 mtu 9000 up
```

Repeat the same `oratcp` performance methodology as described above with the higher MTU size to see if greater throughput is achieved.

If performance gains are noticed, work with system and network engineers to change MTU size for DG transport for both primary and standby databases.

Using This Data

The throughput numbers can be used to determine throughput to aid in Redo Transport and Instantiation situations.

Redo Transport

If the single process throughput does not exceed the single instance redo generation rate for a primary database, the standby will not stay current with the primary during these times. Further evaluation and network tuning by the network engineering team may be required in these cases.

Instantiation

Once the maximum aggregate throughput of all nodes is understood, a rough estimate for instantiation can be developed. As an example, if there is a 100 TB database on a 2-node RAC to be instantiated and each node can achieve 300 MB/s it should take about 50 hours to copy the data files. Additional work to instantiate will add some percentage to that number (~30%).

```
300 MB/s * 60 seconds/minute * 60 minutes/hour * 2 nodes = ~2 TB/hr aggregate
for both nodes
100TB / 2TB/hr = ~50 hours
```

The steps to instantiate a database using large database optimizations such as using multiple nodes is described in Creating a Physical Standby database using RMAN restore database from service (Doc ID 2283978.1).

Determining Oracle Data Guard Protection Mode

Oracle Data Guard can run in three different protection modes, which cater to different performance, availability, and data loss requirements. Use this guide to determine which protection mode fits your business requirements and your potential environmental constraints.

Maximum Protection mode guarantees that no data loss will occur if the primary database fails, even in the case of multiple failures (for example, the network between the primary and standby fails, and then at a later time, the primary fails). This policy is enforced by never signaling commit success for a primary database transaction until at least one synchronous Data Guard standby has acknowledged that redo has been hardened to disk. Without such an acknowledgment the primary database will stall and eventually shut down rather than allow unprotected transactions to commit.

To maintain availability in cases where the primary database is operational but the standby database is not, the best practice is to always have a minimum of **two synchronous standby databases** in a Maximum Protection configuration. Primary database availability is not impacted if it receives acknowledgment from at least one synchronous standby database.

Choose this protection mode if zero data loss is more important than database availability. Workload impact analysis is recommended to measure whether any overhead is acceptable when enabling `SYNC` transport.

Maximum Availability mode guarantees that no data loss will occur in cases where the primary database experiences the first failure to impact the configuration. Unlike the Maximum Protection mode, Maximum Availability will wait a maximum of `NET_TIMEOUT` seconds for an acknowledgment from any of the standby databases, after which it will signal commit success to the application and move to the next transaction. Primary database availability (thus the name of the mode) is not impacted by an inability to communicate with the standby (for

example, due to standby or network outages). Data Guard will continue to ping the standby and automatically re-establish connection and resynchronize the standby database when possible, but during the period when primary and standby have diverged there will be data loss should a second failure impact the primary database.

For this reason, it is a best practice to monitor protection level, which is simplest using Enterprise Manager Grid Control, and quickly resolve any disruption in communication between the primary and standby before a second failure can occur. This is the most common zero data loss database protection mode.

Choose this protection mode if zero data loss is very important but you want the primary database to continue to be available even with the unlikely case that all standby databases are not reachable. You can complement this solution by integrating multiple standby databases or using Far Sync instances to implement a zero data loss standby solution across a WAN. Workload impact analysis is recommended to measure whether any overhead is acceptable when enabling `SYNC` transport.

Maximum Performance mode is the default Data Guard mode, and it provides the highest level of data protection that is possible without affecting the performance or the availability of the primary database. This is accomplished by allowing a transaction to commit as soon as the redo data needed to recover that transaction is written to the local online redo log at the primary database (the same behavior as if there were no standby database). Data Guard transmits redo concurrently to 1) the standby database directly from the primary log buffer and 2) to the local online redo log write asynchronously enabling a very low potential data loss if the primary site is lost. There is never any wait for standby acknowledgment but the potential data loss for this data protection mode can still be near zero..

Similar to Maximum Availability mode, it is a best practice to monitor the protection level using Enterprise Manager Grid Control, and quickly resolve any disruption in communication between primary and standby before a second failure can occur.

Choose this mode if minimum data loss is acceptable and zero performance impact on the primary is required.

Offloading Queries to a Read-Only Standby Database

Offloading queries and reporting workloads to read-only standby databases can free up your primary database system resources, giving you the ability to add more users, workloads, or even databases.

When you leverage both primary and standby database resources, your business and your applications benefit with higher total system usage, and potentially higher application throughput.

Offload appropriate workloads by following these steps.

1. Identify which application modules are read-only or read-mostly.
    - Evaluate whether you have application services or modules that are read-only.
    - Small and short read-only queries are good candidates to offload to the standby database.
    - Short DMLs, especially those that are response-time sensitive, should not be offloaded to the standby.
    - Large reports or analytic reports are good candidates to offload.
    - Reports that are primarily reads, and that may have an infrequent DML, typically at the start or end of a report, may be good candidates to offload.

To enable DML Redirection, see ADG_REDIRECT_DML.

2. Gather information about the expected application performance, throughput, response time, or elapsed time service levels for each offload candidate.
   - Once you have determined which queries and reports are good candidates to offload, find out the required expected and maximum response time or elapsed time for each of them. For example some large analytic reports must complete within a 2 hour time span.
   - For short queries, determine the expected response time and throughput expectations.
   - These requirements are sometimes referred to as application performance Service Level Agreements, which you need for the next step.
3. Test the performance of each candidate on the standby, and determine whether it meets your requirements.
   - Even though the primary and standby databases have essentially identical data, they are independent databases, independent machines, independent configurations, and have different workloads. For example, an Active Data Guard read-only standby database has a redo apply workload plus the queries that are offloaded, while the primary database may have OLTP, batch, and query workloads.
   - Reported elapsed times, query response time, and workload performance may vary between the primary and standby due to these system, configuration, and workload differences.
   - Tuning requires that you understand system resources, SQL plans, and individual query CPU and wait profile. The tuning recommendations are applicable for both primary and standby databases. See Diagnosing and Tuning Database Performance .
4. Offload a subset of the queries that meet your performance requirements, freeing up resources on the primary database for additional processing capacity.
   - Once you have determined which queries and reports can be offloaded, and the performance of those activities are acceptable, then slowly offload some of the workload and monitor it.
   - Do not oversubscribe and offload too much workload to the standby such that redo apply cannot keep pace after tuning. If the standby falls behind. then you lose that standby as a viable role transition target, and in most cases a standby that lags cannot be used to offload queries.

What if a specific query does not meet your requirements?

1. Consult with a performance engineer and follow the recommendations in Database Performance Tuning Guide.
2. A particular query response time or throughput or report elapsed time is not guaranteed to be the same on the standby system as it was on the primary. Analyze the system resources, SQL plans, overall CPU work time and wait times.

   For example, you may see `standby query scn advance wait` is contributing to a much longer elapsed time in one of your short queries. This wait increase is attributed to Active Data Guard redo apply. If a query sees a certain row in a data block and needs to roll it back because the transaction has not committed as of the query System Commit Number (SCN), it needs to apply corresponding undo to get a consistent read for that query. If the redo for the corresponding undo change has not been applied by redo apply yet, the query needs to wait. The presence of such wait is itself not an issue, and typically may be a couple of milliseconds, but it will vary by workload and may be higher in Real Application Cluster database systems.

15

Configure and Deploy Oracle Data Guard

Use the following Oracle MAA best practice recommendations to configure and deploy Oracle Data Guard.

Oracle Data Guard Configuration Best Practices

The following topics describe Oracle MAA best practices for configuring your Oracle Data Guard configuration.

Apply Oracle Database Configuration Best Practices First

Before you implement the Oracle Data Guard best practices that follow, apply the Oracle Database configuration best practices.

The Oracle Data Guard configuration best practices are considered additional to the general Oracle Database configuration best practices, and will help you achieve the services levels you expect of the MAA Gold reference architecture. It is implied that all of the database configuration best practices should be followed in a Data Guard configuration, and that the Data Guard recommendations discussed here supplant the general database recommendation where there are conflicts.

See Oracle Database Configuration Best Practices for more details.

Use Recovery Manager to Create Standby Databases

There are several methods you can use to create an Oracle Data Guard standby database, but because of its simplicity, the Oracle MAA recommended approach is to create a physical standby database using the `RMAN RESTORE ... FROM SERVICE` clause.

For information about this approach see Creating a Physical Standby database using RMAN restore from service (Doc ID 2283978.1).

Use Oracle Data Guard Broker with Oracle Data Guard

Use Oracle Data Guard broker to create, manage, and monitor an Oracle Data Guard configuration.

You can perform all Data Guard management operations locally or remotely using the broker interfaces: the Data Guard management pages in Oracle Enterprise Manager, which is the broker's graphical user interface (GUI), and the Data Guard command-line interface, called DGMGRL.

The broker interfaces improve usability and centralize management and monitoring of the Data Guard configuration. Available as a feature of Oracle Database Enterprise Edition and Personal Edition, the broker is also integrated with Oracle Database, Oracle Enterprise Manager, and Oracle Cloud Control Plane.

Example Broker Installation and Configuration

The following is an example broker installation and configuration, which is used in all of the broker configuration best practices examples.

Prerequisites:

- Primary database, standby database, and observers reside on separate servers and hardware to provide fault isolation.
- Both primary and standby databases must use an `SPFILE`.
- Set the `DG_BROKER_START` initialization parameter to `TRUE`.
- If any of the databases in the configuration is an Oracle RAC database, you must set up the `DG_BROKER_CONFIG_FILE`*n* initialization parameters for that database such that they point to the same shared files for all instances of that database. The shared files could be files on a cluster file system, if available, on raw devices, or stored using Oracle Automatic Storage Management.

1. If they do not already exist, create Oracle Net Services aliases that connect to the primary and the standby databases. These aliases should exist in the database home for each host or member of the Data Guard configuration. For Oracle RAC configurations, the aliases should connect using the SCAN name.

```
chicago =
 (DESCRIPTION =
  (ADDRESS_LIST =
    (ADDRESS=(PROTOCOL= TCP)
     (HOST=prmy-scan)(PORT=1521)))
 (CONNECT_DATA =
   (SERVER = DEDICATED)
    (SERVICE_NAME = chicago)))

boston =
 (DESCRIPTION =
  (ADDRESS_LIST =
    (ADDRESS=(PROTOCOL= TCP)
     (HOST=stby-scan)(PORT=1521)))
 (CONNECT_DATA =
   (SERVER = DEDICATED)
    (SERVICE_NAME = boston)))
```

2. On a primary host, connect with DGMGRL and create the configuration.

```
$ dgmgrl sys
Enter password: password
DGMGRL> create configuration 'dg_config' as  primary database is 'chicago'
connect identifier is chicago;

 Configuration "dg_config" created with primary database "chicago"

DGMGRL> add database 'boston' as connect identifier is boston;

 Database "boston" added
```

```
DGMGRL> enable configuration;
 Enabled.
```

3. By default the broker sets up a LOG_ARCHIVE_DEST_*n* for Maximum Performance database protection mode.

   The broker configures the remote archive destinations with the default values for asynchronous transport, as shown here.

```
log_archive_dest_3=service="boston", ASYNC NOAFFIRM delay=0 optional
compression=disable max_failure=0 reopen=300 db_unique_name="boston"
net_timeout=30, valid_for=(online_logfile,all_roles)
```

Configure Redo Transport Mode

Configure the redo transport service on each configuration member by setting the LogXptMode property to one of the following modes.

- **ASYNC** configures redo transport services for this standby database using the ASYNC and NOAFFIRM attributes of the LOG_ARCHIVE_DEST_*n* initialization parameter. This mode, along with standby redo log files, enables minimum data loss data protection of potentially less couple seconds with zero performance impact.
- **FASTSYNC** configures redo transport services for this standby database using the SYNC and NOAFFIRM attributes of the LOG_ARCHIVE_DEST_*n* initialization parameter. Configure synchronous redo transport mode with the NOAFFIRM attribute (default=AFFIRM) when using maximum availability mode protection mode. This helps to minimize the performance impact of synchronous redo transport by acknowledging the receipt of redo once it has been successfully received and verified within standby memory, but before the redo has been written to the standby redo log. Zero data loss protection is still preserved when only the primary database fails.
- **SYNC** configures redo transport services for this standby database using the SYNC and AFFIRM attributes of the LOG_ARCHIVE_DEST_*n* initialization parameter. This mode, along with standby redo log files, is required for configurations operating in either maximum protection mode or maximum availability mode. This redo transport service enables zero data loss data protection to the primary database, but also can incur a higher performance impact if the round trip latency between primary and standby is high (for example, more than 2ms). This option is required for maximum protection mode.

Use the EDIT DATABASE SET PROPERTY command to set the transport mode the broker configuration, as shown in these examples.

```
DGMGRL> EDIT DATABASE 'boston' SET PROPERTY LogXptMode=ASYNC;

DGMGRL> EDIT DATABASE 'chicago' SET PROPERTY LogXptMode=FASTSYNC;

DGMGRL> EDIT DATABASE 'SanFran' SET PROPERTY LogXptMode=SYNC;
```

Validate the Broker Configuration

To identify any problems with the overall configuration, validate it using the following steps.

1. Show the status of the broker configuration using the SHOW CONFIGURATION command.

```
DGMGRL> show configuration;

 Configuration - dg
```

```
  Protection Mode: MaxPerformance
  Members:
  chicago - Primary database
  boston - Physical standby database

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS   (status updated 18 seconds ago)
```

If the configuration status is `SUCCESS`, everything in the broker configuration is working properly. However, if the configuration status is `WARNING` or `ERROR` then something is wrong in the configuration. Additional error messages that accompany a `WARNING` or `ERROR` status can be used to identify the issues. The next step is to examine each database in the configuration to narrow down what the specific error is related to.

2. To identify warnings on the primary and standby databases, show their statuses using the `SHOW DATABASE` command.

```
DGMGRL> show database chicago

Database - chicago

  Role:               PRIMARY
  Intended State:     TRANSPORT-ON
  Instance(s):
    tin1
    tin2

Database Status:
SUCCESS
```

If the database status is `SUCCESS` then the database is working properly. However, if database status is `WARNING` or `ERROR`, then something is wrong in the database. Additional error messages accompany the `WARNING` or `ERROR` status and can be used to identify current issues.

Repeat the `SHOW DATABASE` command on the standby database and assess any error messages.

3. Validate the databases on Oracle Database 12.1 and later.

In addition to the above commands, in Oracle Database 12.1 and later, the Data Guard broker features a `VALIDATE DATABASE` command.

```
DGMGRL> validate database chicago

  Database Role:    Primary database
   Ready for Switchover:  Yes

DGMGRL> validate database boston;

   Database Role:     Physical standby database
   Primary Database:  tin
```

```
Ready for Switchover:  No
Ready for Failover:    Yes (Primary Running)

Capacity Information:
  Database  Instances        Threads
  tin       2                2
  can       1                2
  Warning: the target standby has fewer instances than the
  primary database, this may impact application performance

Standby Apply-Related Information:
  Apply State:      Not Running
  Apply Lag:        Unknown
  Apply Delay:      0 minutes
```

The `VALIDATE DATABASE` command does not provide a `SUCCESS` or `WARNING` status and must be examined to determine if any action needs to be taken.

Configure Fast Start Failover

Fast-start failover allows the broker to automatically fail over to a previously chosen standby database in the event of loss of the primary database. Enabling fast-start failover is requirement to meet stringent RTO requirements in the case of primary database, cluster, or site failure.

Fast-start failover quickly and reliably fails over the target standby database to the primary database role, without requiring you to perform any manual steps to invoke the failover. Fast-start failover can be used only in a broker configuration.

If the primary database has multiple standby databases, then you can specify multiple fast-start failover targets, using the `FastStartFailoverTarget` property. The targets are referred to as candidate targets. The broker selects a target based on the order in which they are specified on the `FastStartFailoverTarget` property. If the designated fast-start failover target develops a problem and cannot be the target of a failover, then the broker automatically changes the fast-start failover target to one of the other candidate targets.

You can use any protection mode with fast-start failover. The maximum protection and maximum availability modes provide an automatic failover environment guaranteed to lose no data. Maximum performance mode provides an automatic failover environment guaranteed to lose no more than the amount of data (in seconds) specified by the `FastStartFailoverLagLimit` configuration property. This property indicates the maximum amount of data loss that is permissible in order for an automatic failover to occur. It is only used when fast-start failover is enabled and the configuration is operating in maximum performance mode.

1. Set the `FastStartFailoverThreshold` property to specify the number of seconds you want the observer and target standby database to wait, after detecting the primary database is unavailable, before initiating a failover, as shown in this example.

   ```
   DGMGRL> EDIT CONFIGURATION SET PROPERTY FastStartFailoverThreshold =
   seconds;
   ```

   A fast-start failover occurs when the observer and the standby database both lose contact with the production database for a period of time that exceeds the value set for `FastStartFailoverThreshold`, and when both parties agree that the state of the

configuration is synchronized (Maximum Availability), or that the lag is not more than the configured `FastStartFailoverLagLimit` (Maximum Performance).

An optimum value for `FastStartFailoverThreshold` weighs the trade-off between the fastest possible failover (minimizing downtime) and unnecessarily triggering failover because of temporary network irregularities or other short-lived events that do not have material impact on availability.

The default value for `FastStartFailoverThreshold` is 30 seconds.

The following table shows the recommended settings for `FastStartFailoverThreshold` in different use cases.

Table 15-1 Minimum Recommended Settings for FastStartFailoverThreshold

Configuration	minimum Recommended Setting
Single-instance primary, low latency, and a reliable network	15 seconds
Single-instance primary and a high latency network over WAN	30 seconds
Oracle RAC primary	Oracle RAC miscount + reconfiguration time + 30 seconds

2. Determine where to place the observer in your topology.

   In an ideal state fast-start failover is deployed with the primary, standby, and observer, each within their own availability domain (AD) or data center; however, configurations that only use two availability domains, or even a single availability domain, must be supported. The following are observer placement recommendations for two use cases.

   Deployment Configuration 1: 2 regions with two ADs in each region.

   - Initial primary region has the primary database in AD1, and two high availability observers (one observer in AD2 and second HA observer in AD1)
   - Initial standby region has the standby database in AD1, and two high availability observers used after role change (one observer in AD2 and second HA observer in AD1)
   - For the observer, MAA recommends at least 2 observer targets in the same primary region but in different ADs

   Deployment Configuration 2: 2 regions with only 1 AD in each region

   - Initial primary regions have the primary database and two light weight servers to host observers
   - Initial standby region has the standby database and two light weight servers to host observers (when there is a role change)

3. Configure observer high availability.

   You can register up to three observers to monitor a single Data Guard broker configuration. Each observer is identified by a name that you supply when you issue the `START OBSERVER` command. You can also start the observers as a background process.

```
DGMGRL> sys@boston
Enter password: password
DGMGRL> start observer number_one in background;
```

On the same host or a different host you can start additional observers for high availability:

```
DGMGRL> sys@boston
Enter password: password
DGMGRL> start observer number_two in background;
```

Only the primary observer can coordinate fast-start failover with Data Guard broker. All other registered observers are considered to be backup observers.

If the observer was not placed in the background then the observer is a continuously running process that is created when the `START OBSERVER` command is issued. Therefore, the command-line prompt on the observer computer does not return until you issue the `STOP OBSERVER` command from another `DGMGRL` session. To issue commands and interact with the broker configuration, you must connect using another `DGMGRL` client session.

Now that you have correctly configured fast-start failover, the following conditions can trigger a failover.

- Database failure where all database instances are down
- Data files taken offline because of I/O errors
- Both the Observer and the standby database lose their network connection to the production database, and the standby database confirms that it is in a synchronized state
- A user-configurable condition

Optionally, you can specify the following conditions for which a fast-start failover can be invoked. It is recommend that you leave these user-configurable conditions at the default values and not invoke an automatic failover.

- Data file offline (write error)
- Corrupted Dictionary
- Corrupted Control file
- Inaccessible Log file
- Stuck Archiver
- ORA-240 (control file enqueue timeout)

Should one of these conditions be detected, the observer fails over to the standby, and the primary shuts down, regardless of how `FastStartFailoverPmyShutdown` is set. Note that the for user-configurable conditions, the fast-start failover threshold is ignored and the failover proceeds immediately.

Fast Start Failover with Multiple Standby Databases

The `FastStartFailoverTarget` configuration property specifies the `DB_UNIQUE_NAME` of one or more standby databases that can act as target databases in a fast-start failover situation when the database on which the property is set is the primary database. These possible target databases are referred to as candidate fast-start failover targets.

The `FastStartFailoverTarget` configuration property can only be set to the name of physical standbys. It cannot be set to the name of a snapshot standby database, far sync instance, or Zero Data Loss Recovery Appliance.

If only one physical standby database exists, then the broker selects that as the default value for this property on the primary database when fast-start failover is enabled. If more than one

physical standby database exists, then the broker selects one based on the order in which they are specified in the property definition. Targets are verified when fast-start failover is enabled

Set Log Buffer Optimally

Set `LOG_BUFFER` to a minimum of 256 MB when using Oracle Data Guard with asynchronous redo transport.

Doing so allows the asynchronous redo transport to read redo from the log buffer and avoid disk I/Os to online redo logs. For workloads that have a very high redo generation rate (for example, > 50 MB/sec per database instance) the `LOG_BUFFER` can be increased up to maximum value allowed for the platform being used.

> **Note:**
>
> The maximum `LOG_BUFFER` setting for Linux platform is 2 GB and for Windows is 1 GB.

Set Send and Receive Buffer Sizes

Redo transport processes, especially the receive/standby side, generally benefit from more TCP socket buffers in high latency network paths. TCP socket buffers can be managed by the TCP stack dynamically.

Setting the maximum values for tcp_rmem and tcp_wmem allows the kernel to dynamically modify the buffers allocated to a socket as needed.

Bandwidth Delay Product (BDP) is the product of the network link capacity of a channel and the round time, or latency. The minimum recommended value for socket buffer sizes is 3*BDP, especially for a high-latency, high-bandwidth network. Use oratcptest to tune the socket buffer sizes.

As root, set the maximum value for tcp_rmem and tcp_wmem to 3*<Bandwidth Delay Product>. In this example BDP is 16MB. The other two values for these parameters should be left as they currently are set on the system.

```
# sysctl -w net.ipv4.tcp_rmem='4096 87380 16777216';

# sysctl -w net.ipv4.tcp_wmem='4096 16384 16777216';
```

Using sysctl to change these values changes them dynamically in memory only and will be lost when the system is rebooted. Additionally set these values in /etc/sysctl.conf on linux systems. Add these entries if the values are absent in the file.

```
net.ipv4.tcp_rmem='4096 87380 16777216'

net.ipv4.tcp_wmem='4096 16384 16777216'
```

Set SDU Size to 65535 for Synchronous Transport Only

With Oracle Net Services you can control data transfer by adjusting the session data unit (SDU) size. Oracle testing has shown that setting the SDU parameter to its maximum value of 65535 improves performance of synchronous transport.

You can set SDU on a per connection basis using the SDU parameter in the local naming configuration file, `tnsnames.ora`, and the listener configuration file, `listener.ora`, or you can set the SDU for all Oracle Net Services connections with the profile parameter `DEFAULT_SDU_SIZE` in the `sqlnet.ora` file.

Configure Online Redo Logs Appropriately

Redo log switching has a significant impact on redo transport and apply performance. Follow these best practices for sizing the online redo logs on the primary and standby databases.

Following these guidelines for online redo logs.

- All online redo log groups should have identically sized logs (to the byte).
- Online redo logs should reside on high performing disks (DATA disk groups).
- Create a minimum of three online redo log groups per thread of redo on Oracle RAC instances.
- Create online redo log groups on shared disks in an Oracle RAC environment.
- Multiplex online redo logs (multiple members per log group) unless they are placed on high redundancy disk groups.
- Size online redo logs to switch no more than 12 times per hour (every ~5 minutes). In most cases a log switch every 15 to 20 minutes is optimal even during peak workloads.

Sizing Redo Logs

Size the redo logs based on the peak redo generation rate of the primary database.

You can determine the peak rate by running the query below for a period of time that includes the peak workload. The peak rate could be seen at month-end, quarter-end, or annually. Size the redo logs to handle the highest rate in order for redo apply to perform consistently during these workloads.

```
SQL> SELECT thread#,sequence#,blocks*block_size/1024/1024 MB,(next_time-
first_time)*86400 sec,
 blocks*block_size/1024/1024)/((next_time-first_time)*86400) "MB/s"
 FROM v$archived_log WHERE ((next_time-first_time)*86400<>0) and first_time
 between to_date('2015/01/15 08:00:00','YYYY/MM/DD HH24:MI:SS')
 and to_date('2015/01/15 11:00:00','YYYY/MM/DD HH24:MI:SS') and dest_id=1
order by first_time;

   THREAD#  SEQUENCE#         MB        SEC       MB/s
---------- ---------- ---------- ---------- ----------
         2       2291 29366.1963        831  35.338383
         1       2565 29365.6553        781 37.6000708
         2       2292 29359.3403        537  54.672887
         1       2566 29407.8296        813 36.1719921
         2       2293 29389.7012        678 43.3476418
         2       2294 29325.2217       1236 23.7259075
         1       2567 11407.3379       2658 4.29169973
         2       2295 29452.4648        477 61.7452093
         2       2296 29359.4458        954 30.7751004
         2       2297 29311.3638        586 50.0193921
         1       2568 3867.44092       5510 .701894903
```

Choose the redo log size based on the peak generation rate with the following chart.

Table 15-2 Recommended Redo Log Size

Peak Redo Rate	Recommended Redo Log Size
<= 1 MB/s	1 GB
<= 5 MB/s	4 GB
<= 25 MB/s	16 GB
<= 50 MB/s	32 GB
> 50 MB/s	64 GB

Use Standby Redo Log Groups

Configure the standby redo log groups on all primary and standby databases for improved availability and performance.

For each redo log thread--a thread is associated with an Oracle RAC database instance--the number of standby redo log groups must be greater than or equal to (>=) the number of online redo log groups.

Consider the following additional guidelines when creating standby redo log groups.

- All online redo logs and standby redo log groups should have identically sized logs (to the byte). Standby redo logs are not used if they are not the same size as the online redo logs.
- All standby redo log groups should have identically sized logs (to the byte) on both the primary and standby databases.
- Standby redo logs should reside on high performing disks (DATA disk group).
- Standby redo logs should be multiplexed (multiple members per log group) unless placed on high redundancy disk groups. Multiplexing standby redo logs is optional in all cases because Data Guard can fetch any missing redo.
- In an Oracle RAC environment, create standby redo logs on a shared disk.
- In an Oracle RAC environment, assign a thread to each standby redo log group.

The following example creates three log groups for each redo thread.

```
SQL> ALTER DATABASE ADD STANDBY LOGFILE THREAD 1 GROUP 7 ('+DATA') SIZE
4194304000, GROUP 8 ('+DATA') SIZE 4194304000, GROUP 9 ('+DATA') SIZE
4194304000;

SQL> ALTER DATABASE ADD STANDBY LOGFILE THREAD 2 GROUP 10 ('+DATA') SIZE
4194304000, GROUP 11 ('+DATA') SIZE 4194304000, GROUP 12 ('+DATA') SIZE
419430400
```

To check the thread number and group numbers of the online redo logs, query the `V$LOG` view.

```
SQL> SELECT * FROM V$LOG;
```

To check the results of the `ALTER DATABASE ADD STANDBY LOGFILE THREAD` statements, query the `V$STANDBY_LOG` view.

```
SQL> SELECT * FROM V$STANDBY_LOG;
```

Protect Against Data Corruption

Oracle Database corruption prevention, detection, and repair capabilities are built on internal knowledge of the data and transactions it protects, and on the intelligent integration of its comprehensive high availability solutions.

When data corruption is detected, Oracle Data Guard, block media recovery, and data file media recovery can recover the data. Database-wide logical corruptions caused by human or application errors can be undone with Oracle Flashback Technologies.

Tools are also available for proactive validation of logical data structures. For example, the SQL*Plus `ANALYZE TABLE` statement detects inter-block corruptions.

Achieve the most comprehensive data corruption prevention and detection with these best practices.

- Use Oracle Data Guard with physical standby databases to prevent widespread block corruption. Oracle Data Guard is the best solution for protecting Oracle data against data loss and corruption, and lost writes.
- Set the Oracle Database block-corruption initialization parameters on the Data Guard primary and standby databases as shown in the following table.

Table 15-3 Block-Corruption Initialization Parameter Settings

On the primary database set...	On the standby databases set...
`DB_BLOCK_CHECKSUM=MEDIUM` or `FULL`	`DB_BLOCK_CHECKSUM=MEDIUM` or `FULL`
`DB_LOST_WRITE_PROTECT=TYPICAL`	`DB_LOST_WRITE_PROTECT=TYPICAL`
`DB_BLOCK_CHECKING=FALSE*`	`DB_BLOCK_CHECKING=MEDIUM` or `FULL`

* `DB_BLOCK_CHECKING` on the `PRIMARY` is recommended to be set to `MEDIUM` or `FULL` but only after a full performance evaluation with the application.

- Performance overhead is incurred on every block change, therefore performance testing is of particular importance when setting the `DB_BLOCK_CHECKING` parameter. Oracle highly recommends the minimum setting of `DB_BLOCK_CHECKING=MEDIUM` (block checks on data blocks but not index blocks) on either the primary or standby database. If the performance overhead of enabling `DB_BLOCK_CHECKING` to `MEDIUM` or `FULL` is unacceptable on your primary database, then set `DB_BLOCK_CHECKING` to `MEDIUM` or `FULL` for your standby databases.

The following recommendations also help to protect against data corruptions.

- Use Oracle Automatic Storage Management (Oracle ASM) to provide disk mirroring to protect against disk failures.
- Use Oracle ASM `HIGH REDUNDANCY` for optimal corruption repair. Using Oracle ASM redundancy for disk groups provides mirrored extents that can be used by the database if an I/O error or corruption is encountered. For continued protection, Oracle ASM redundancy provides the ability to move an extent to a different area on a disk if an I/O error occurs. The Oracle ASM redundancy mechanism is useful if you have bad sectors returning media errors.
- Enable Flashback Technologies for fast point-in-time recovery from logical corruptions most often caused by human error and for fast reinstatement of a primary database following failover.

- Use RMAN for additional block checks during backup and restore operations. Implement a backup and recovery strategy with Recovery Manager (RMAN) and periodically use the RMAN `BACKUP VALIDATE CHECK LOGICAL` scan to detect corruptions.
- Use Zero Data Loss Recovery Appliance for backup and recovery validation including corruption checks and repairs, central backup validation, reduced production database impact, and Enterprise Cloud backup and recovery solution.

Use Flashback Database for Reinstatement After Failover

Enable Flashback Database on both the primary and standby database, so that if the original primary database has not been damaged, you can reinstate the original primary database as a new standby database following a failover.

If there is a failure during the switchover process, then it can easily be reversed when Flashback Database is enabled.

Set `DB_FLASHBACK_RETENTION_TARGET` to the same value on the standby database as the primary. Set `DB_FLASHBACK_RETENTION_TARGET` initialization parameter to the largest value prescribed by any of the following conditions that apply.

- To leverage flashback database to reinstate your failed primary database after Data Guard failover, for most cases set `DB_FLASHBACK_RETENTION_TARGET` to a minimum of 120 (minutes) to enable reinstatement of a failed primary.
- If using Flashback Database for fast point in time recovery from user error or logical corruptions, set `DB_FLASHBACK_RETENTION_TARGET` to a value equal to the farthest time in the past to which the database should be recovered. If you can detect and repair from logical corruptions in less than 24 hours, then set `DB_FLASHBACK_RETENTION_TARGET` to a minimum of 1440 (minutes).

Use Force Logging Mode

When the primary database is in `FORCE LOGGING` mode, all database data changes are logged. `FORCE LOGGING` mode ensures that the standby database remains consistent with the primary database.

If it is not possible to use this mode because you require the load performance with `NOLOGGING` operations, then see Enable an Appropriate Logging Mode for other options.

You can enable force logging immediately by issuing an `ALTER DATABASE FORCE LOGGING` statement. If you specify `FORCE LOGGING`, then Oracle waits for all ongoing non-logged operations to finish.

Configure Fast Start Failover to Bound RTO and RPO (MAA Gold Requirement)

Enabling fast-start failover is requirement to meet stringent RTO requirements in the case of primary database, cluster, or site failure. With Data Guard fast-start failover, there's a Data Guard observer to provide quorum of 2 and to preserve database consistency and prevent database split brains.

Fast-start failover allows the Data Guard broker to automatically fail over to a previously chosen standby database in the event of loss of the primary database. Fast-start failover quickly and reliably switches the target standby database over to the primary database role,

without requiring you to perform any manual steps to invoke the failover. Fast-start failover can be used only in a Data Guard broker configuration.

If the primary database has multiple standby databases, then you can specify multiple fast-start failover targets, using the `FastStartFailoverTarget` property. The targets are referred to as candidate targets. The broker selects a target based on the order in which they are specified on the `FastStartFailoverTarget` property. If the designated fast-start failover target develops a problem and cannot be the target of a failover, then the broker automatically changes the fast-start failover target to one of the other candidate targets.

You can use any Data Guard protection mode with fast-start failover. The maximum protection and maximum availability modes provide an automatic failover environment guaranteed to lose no data. Maximum performance mode provides an automatic failover environment guaranteed to lose no more than the amount of data (in seconds) specified by the `FastStartFailoverLagLimit` configuration property. This property indicates the maximum amount of data loss that is permissible in order for an automatic failover to occur. It is only used when fast-start failover is enabled and the configuration is operating in maximum performance mode.

1. Set the `FastStartFailoverThreshold` property to specify the number of seconds you want the observer and target standby database to wait, after detecting the primary database is unavailable, before initiating a failover, as shown in this example.

   ```
   DGMGRL> EDIT CONFIGURATION SET PROPERTY FastStartFailoverThreshold =
   seconds;
   ```

   A fast-start failover occurs when the observer and the standby database both lose contact with the production database for a period of time that exceeds the value set for `FastStartFailoverThreshold`, and when both parties agree that the state of the configuration is synchronized (Maximum Availability mode), or that the lag is not more than the configured `FastStartFailoverLagLimit` (Maximum Performance mode).

   An optimum value for `FastStartFailoverThreshold` weighs the trade-off between the fastest possible failover (minimizing downtime) and unnecessarily triggering failover because of temporary network irregularities or other short-lived events that do not have material impact on availability.

   The default value for `FastStartFailoverThreshold` is 30 seconds.

   The following table shows the recommended settings for `FastStartFailoverThreshold` in different use cases.

Configuration	Minimum Recommended Setting
Single-instance primary, low latency, and a reliable network	15 seconds
Single-instance primary and a high latency network over WAN	30 seconds
Oracle RAC primary	Oracle RAC miscount + reconfiguration time + 30 seconds For Exadata systems, minimum setting of 30 seconds

2. Determine where to place the observer in your topology.

   The Data Guard broker observer provides a quorum of 2 to preserve database consistency and avoid split brains. Data Guard fast-start failover always guarantees that only one primary database exists, and external consistency is guaranteed by routing transactions to

the primary database. In an ideal state, fast-start failover is deployed with the primary, standby, and observer, each within their own availability domain (AD) or data center; however, configurations that only use two availability domains, or even a single availability domain, must be supported. The following are observer placement recommendations for two use cases.

- **Deployment Configuration 1: 2 regions with two ADs in each region.**
  - Initial primary region has the primary database in AD1, and two high availability observers (one observer in AD2 and second HA observer in AD1)
  - Initial standby region has the standby database in AD1, and two high availability observers used after role change (one observer in AD2 and second HA observer in AD1)
  - For the observer, MAA recommends at least 2 observer targets in the same primary region but in different ADs
- **Deployment Configuration 2: 2 regions with only 1 AD in each region**
  - Initial primary regions have the primary database and two light weight servers to host observers
  - Initial standby region has the standby database and two light weight servers to host observers (when there is a role change)

3. Configure observer high availability.

   You can register up to three observers to monitor a single Data Guard broker configuration. Each observer is identified by a name that you supply when you issue the `START OBSERVER` command. You can also start the observers as a background process.

```
DGMGRL> sys@boston
Enter password:
DGMGRL> start observer number_one in background;
```

   On the same host or a different host you can start additional observers for high availability:

```
DGMGRL> sys@boston
Enter password:
DGMGRL> start observer number_two in background;
```

   Only the primary observer can coordinate fast-start failover with Data Guard broker. All other registered observers are considered to be backup observers.

   If the observer was not placed in the background, then the observer is a continuously executing process that is created when the `START OBSERVER` command is issued. Therefore, the command-line prompt on the observer computer does not return until you issue the `STOP OBSERVER` command from another `DGMGRL` session. To issue commands and interact with the broker configuration, you must connect using another `DGMGRL` client session.

Triggering Fast-Start Failover

Now that you have correctly configured fast-start failover, the following conditions can trigger a failover.

- Database failure where all database instances are down
- Datafiles taken offline because of I/O errors

- Both the Observer and the standby database lose their network connection to the production database, and the standby database confirms that it is in a synchronized state
- A user-configurable condition

Optionally, you can specify the following conditions for which a fast-start failover can be invoked. It is recommend that you leave these user-configurable conditions at the default values and not invoke an automatic failover.

- Datafile offline (write error)
- Corrupted Dictionary
- Corrupted Controlfile
- Inaccessible Logfile
- Stuck Archiver
- ORA-240 (control file enqueue timeout)

Should one of these conditions be detected, the observer fails over to the standby, and the primary shuts down, regardless of how `FastStartFailoverPmyShutdown` is set. Note that the for user-configurable conditions, the fast-start failover threshold is ignored and the failover proceeds immediately.

Fast Start Failover with Multiple Standby Databases

The `FastStartFailoverTarget` configuration property specifies the `DB_UNIQUE_NAME` of one or more standby databases that can act as target databases in a fast-start failover situation when the database on which the property is set is the primary database. These possible target databases are referred to as candidate fast-start failover targets.

The `FastStartFailoverTarget` configuration property can only be set to the name of physical standbys. It cannot be set to the name of a snapshot standby database, far sync instance, or Zero Data Loss Recovery Appliance.

If only one physical standby database exists, then the broker selects that as the default value for this property on the primary database when fast-start failover is enabled. If more than one physical standby database exists, then the broker selects one based on the order in which they are specified in the property definition. Targets are verified when fast-start failover is enabled.

Configure Standby AWR

Since Oracle Database 12c (12.2), Automatic Workload Repository (AWR) snapshots can be taken of the standby database.

Standby AWR is the best tool for identifying performance issues with recovery and reporting workloads in an Active Data Guard standby database.

See Managing Automatic Workload Repository in Active Data Guard Standby Databases for details about configuring and managing standby AWR.

> **Note:**
>
> For Oracle Exadata Cloud Data Guard deployments, standby AWR is configured as part of instantiation.

To Create Standby AWR Reports

1. Identify the AWR ID (`NODE_ID`) for the standby database.

   ```
   SQL> select NODE_ID,NODE_NAME from DBA_UMF_REGISTRATION;
   ```

2. Run the reports from the primary database using the `NODE_ID` for the target database as the DBID.
   - For instance level reports (for example, assessing redo apply performance bottlenecks) use the `awrrpti` script.

     ```
     SQL> ?/rdbms/admin/awrrpti
     ```

   - For global AWR reports on the standby (for example, assessing query performance) use the `awrgrpti` script.

     ```
     SQL> ?/rdbms/admin/awrgrpti
     ```

Configuring Multiple Standby Databases

An Oracle Data Guard configuration with multiple standby databases gives you the benefits of both local and remote standby databases.

A local standby database can provide zero data loss failover and application downtime reduced to seconds. If a regional disaster occurs, making the primary and local standby systems inaccessible, the application and database can fail over to the remote standby. See "Gold: Multiple Standby Databases" for a full discussion of the features and benefits of a multiple standby configuration.

Managing Oracle Data Guard Configurations with Multiple Standby Databases

The Oracle Data Guard broker automates management and operation tasks across multiple databases in an Oracle Data Guard configuration. The broker also monitors all of the systems in a single Oracle Data Guard configuration.

In a multi-member Data Guard configuration the following redo transport destinations are supported:

- Oracle Data Guard standby databases
- Far sync instances (See Using Far Sync Instances for more information)
- Oracle Streams downstream capture databases
- Zero Data Loss Recovery Appliance (Recovery Appliance)

Multiple Standby Databases and Redo Routes

You can use the Oracle Data Guard broker `RedoRoutes` property to override the default behavior by which a primary database sends the redo that it generates to every other redo transport destination in the configuration.

An example redo transport topology that differs from the default would be one in which a physical standby, or a far sync instance, forwards redo received from the primary database to one or more destinations, or one in which the redo transport mode used for a given destination is dependent on which database is in the primary role.

Consider a configuration that has a primary database (North_Sales) and two physical standby databases (Local_Sales and Remote_Sales). The Local_Sales database is located in the same data center as the primary for high availability purposes and for simpler application and database failover. The Remote_Sales database is located in a remote data center for disaster recovery purposes.

Rather than have North_Sales ship its redo to both databases, you can use the `RedoRoutes` broker property to configure real-time cascading, in which the local physical standby database forwards to Remote_Sales the redo it receives from North_Sales. To accomplish this, the `RedoRoutes` property is set on North_Sales and Local_Sales as follows:

- On the North_Sales database, the `RedoRoutes` property specifies that if North_Sales is in the primary role, then it should ship redo to the Local_Sales database using synchronous transport mode. This rule prevents the primary from shipping redo data directly to the Remote_Sales database.
- On the Local_Sales database, the `RedoRoutes` property must specify that if North_Sales is in the primary role, then Local_Sales should forward redo it receives from North_Sales on to Remote_Sales.

To see the runtime `RedoRoutes` configuration, use the `SHOW CONFIGURATION` command. For example:

```
DGMGRL> SHOW CONFIGURATION;

Configuration - Sales_Configuration

  Protection Mode: MaxAvailability
  Members:
 North_Sales  - Primary database
    Local_Sales  - Physical standby database
      Remote_Sales  - Physical standby database (receiving current redo)

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS
```

Note that the asynchronous redo transport attribute was explicitly specified in the redo route rule for the Remote_Sales destination to enable real-time cascading of redo to that destination. (Real-time cascading requires a license for the Oracle Active Data Guard option.)

To disable real-time cascading of redo, do not specify the asynchronous redo transport attribute. For example:

```
DGMGRL> EDIT DATABASE 'Local_Sales' SET PROPERTY 'RedoRoutes' =
'(North_Sales : Remote_Sales)';
```

See RedoRoutes for more information.

Using the RedoRoutes Property for Remote Alternate Destinations

The `RedoRoutes` property can be used to set up a remote alternate destination, so that a terminal member can still receive redo data even if the member from which it was receiving the redo data fails.

Using the previous example, you can have the primary database, North_Sales, send redo data directly to Remote_Sales if the Local_Sales standby database failed. It is also possible, using the `PRIORITY` attribute, to specify that once the Local_Sales failure has been resolved it can resume shipping redo to Remote_Sales.

```
DGMGRL> EDIT DATABASE 'North_Sales' SET PROPERTY
 'RedoRoutes' = '(LOCAL : ( Local_Sales ASYNC PRIORITY=1, Remote_Sales ASYNC
PRIORITY=2 ))';
Property "RedoRoutes" updated

DGMGRL> EDIT DATABASE 'Local_Sales'
 SET PROPERTY 'RedoRoutes' = '(North_Sales : Remote_Sales ASYNC)';
Property "RedoRoutes" updated

DGMGRL> SHOW CONFIGURATION;

Configuration - Sales_Configuration

  Protection Mode: MaxPerformance
  Members:
  North_Sales    - Primary database
    Local_Sales  - Physical standby database
      Remote_Sales - Physical standby database
Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS
```

To see the full `RedoRoutes` configuration, use the `SHOW CONFIGURATION VERBOSE` command. For example:

```
DGMGRL> SHOW CONFIGURATION VERBOSE;

Configuration - Sales_Configuration

  Protection Mode: MaxPerformance
  Members:
    North_Sales - Primary database
      Local_Sales - Physical standby database
        Remote_Sales - Physical standby database
      Remote_Sales - Physical standby database (alternate of Local_Sales)

  Properties:
    FastStartFailoverThreshold      = '180'
    OperationTimeout                = '30'
    TraceLevel                      = 'USER'
    FastStartFailoverLagLimit       = '300'
    CommunicationTimeout            = '180'
    ObserverReconnect               = '0'
    FastStartFailoverAutoReinstate  = 'TRUE'
    FastStartFailoverPmyShutdown    = 'TRUE'
    BystandersFollowRoleChange      = 'ALL'
    ObserverOverride                = 'FALSE'
    ExternalDestination1            = ''
    ExternalDestination2            = ''
```

```
    PrimaryLostWriteAction          = 'CONTINUE'
    ConfigurationWideServiceName    = 'c0_CFG'

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS
```

Fast Start Failover with Multiple Standby Databases

The Oracle Data Guard `FastStartFailoverTarget` broker configuration property specifies the `DB_UNIQUE_NAME` of one or more standby databases that can act as target databases in a fast-start failover scenario when the database on which the property is set is the primary database.

These possible target databases are referred to as *candidate fast-start failover targets*. The `FastStartFailoverTarget` property can only be set to the name of physical standbys. It cannot be set to the name of a snapshot standby database, far sync instance, or Zero Data Loss Recovery Appliance.

If only one physical standby database exists, then the broker selects that database as the default value for `FastStartFailoverTarget` on the primary database when fast-start failover is enabled. If more than one physical standby database exists, then the broker selects a single standby based on the order in which they are specified in the property definition. The targets are verified when fast-start failover is enabled.

See also, FastStartFailoverTarget.

Setting FastStartFailoverTarget

If you have two or more standby databases, set up the `FastStartFailoverTarget` configuration property on the primary database to indicate the desired fast-start failover target standby database.

The Oracle Data Guard broker reciprocally sets this property for the target standby database to indicate the primary database as its future target standby database when fast-start failover is actually enabled. There is no need for you set this property on the target standby as this is done for you automatically. For example:

```
DGMGRL> edit database moe set property ='curly,larry';
Property "faststartfailovertarget" updated
```

After `FastStartFailoverTarget` is configured, continue with enabling fast-start failover. When fast-start failover is enabled, you cannot change the `FastStartFailoverTarget` configuration property on the primary or target standby databases.

To change the `FastStartFailoverTarget` property to point to a different standby database, disable fast-start failover, set the `FastStartFailoverTarget` property, and reenable fast-start failover. This action does not impact primary or standby database availability or up time.

Switchover with FastStartFailoverTarget Set

If fast-start failover is enabled with `FastStartFailoverTarget` set you can still perform a switchover or a manual failover, as long the role change is directed to the same standby database that was specified for the `FastStartFailoverTarget` database property on the primary database.

Attempting to switch over to a standby that is not the fast-start failover target results in ORA-16655.

```
DGMGRL> switchover to curly
Performing switchover NOW, please wait...
Error: ORA-16655: specified standby database not the current fast-start
failover target standby
```

To switch over to a standby that is not the primary fast-start target:

1. Disable fast-start failover.

   ```
   DGMGRL> DISABLE FAST_START FAILOVER;
   ```

2. Edit the `FastStartFailoverTarget` property to list the standby you wish to switch over to first.

   ```
   DGMGRL> edit database moe set property
   FastStartFailoverTarget='curly,larry';
   Property "faststartfailovertarget" updated
   ```

3. Enable fast-start failover.

   ```
   DGMGRL> ENABLE FAST_START FAILOVER;
   ```

4. Perform the switchover operation.

   ```
   DGMGRL> switchover to curly
   Performing switchover NOW, please wait...
   ```

Fast-Start Failover Outage Handling

If the primary database's fast-start failover target standby database becomes unavailable, perhaps because the standby database or instance is down or there's an issue with transporting redo, then the primary's fast-start failover target is automatically switched to the next target configured in the `FastStartFailoverTarget` property.

Note that is can take several ping cycles to effect the target switch: one ping to recognize that the current target is not viable, and another ping to propose the target switch and finalize it.

If the original fast-start failover target comes back online, a switch back to the original target is not performed automatically. To get the original target back after an outage you must disable and then enable fast-start failover.

Oracle Active Data Guard Far Sync Solution

To support zero data loss, you can deploy between the primary and standby databases an Oracle Data Guard far sync instance, which is a remote Oracle Data Guard destination that accepts redo from the primary database and then ships that redo to other members of the Oracle Data Guard configuration.

About Far Sync

Far Sync is an Oracle Active Data Guard feature that provides increased flexibility in the location of a disaster recovery site for those who wish to implement zero data loss protection.

Even users who have already deployed Oracle Data Guard synchronous transport can benefit from configuring a far sync instance closer to the primary than their current standby to reduce the performance impact on the production database.

Synchronous redo transport over WAN distances or on an under-performing network often has too large an impact on primary database performance to support zero data loss protection. Oracle Active Data Guard Far Sync provides the ability to perform a zero data loss failover to a remote standby database without requiring a second standby database or complex operation.

Far Sync enables this by deploying a far sync instance (a lightweight Oracle instance) at a distance that is within an acceptable range of the primary for synchronous redo transport. A far sync instance receives redo from the primary using synchronous transport and forwards the redo to up to 29 remote standby databases using asynchronous transport.

Far sync instances are part of the Oracle Active Data Guard Far Sync feature, which requires an Oracle Active Data Guard license.

Offloading to a Far Sync Instance

A far sync instance offloads from the primary any overhead of resolving gaps in redo received by the remote standby database (for example, following network or standby database outages) and can conserve WAN bandwidth by performing redo transport compression without impacting primary database performance.

Note that redo compression requires that the Advanced Compression Option be licensed.

Redo Transport Encryption can additionally be offloaded to the far sync instance. Including Advanced Security Option (ASO) encryption during MAA testing showed no impact to the performance of the primary nor currency of the standby databases.

Oracle recommends using ASO for encryption because it is tested and integrated with Oracle Net and Data Guard.

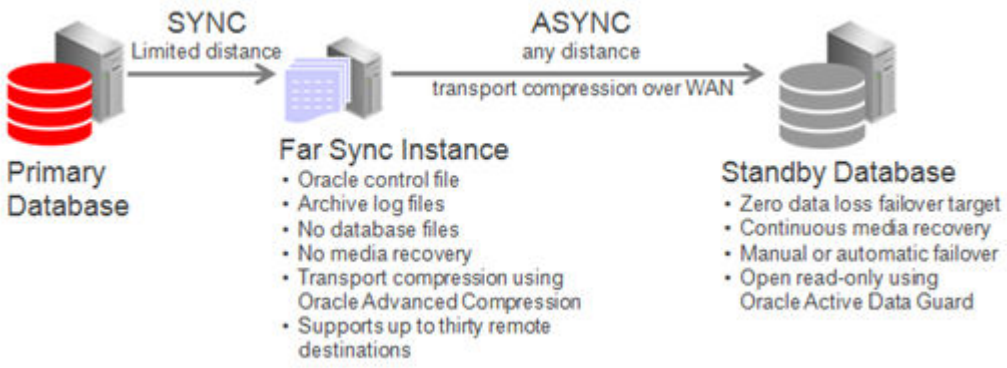
Note that Oracle Advanced Security Option is a licensed option.

Far Sync Deployment Topologies

Oracle Active Data Guard Far Sync provides the ability to perform a zero data loss failover to a remote standby database without requiring a second standby database or complex operation.

Data Guard enables this by deploying a far sync instance (a lightweight Oracle instance that has only a control file, `SPFILE`, password file and standby log files; there are no database files or online redo logs) at a distance that is within an acceptable range of the primary for synchronous transport. A far sync instance receives redo from the primary through synchronous transport and immediately forwards the redo to up to 29 remote standby databases using asynchronous transport. A far sync instance can also forward redo to the new Oracle Database Backup, Logging, and Recovery Appliance.

Figure 15-1 Far Sync Architecture Overview



The following use cases illustrate the benefits of various architecture choices you can implement with far sync instances.

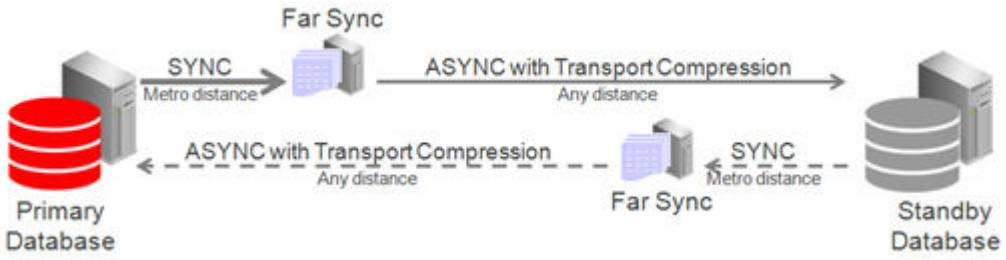
Case 1: Zero Data Loss Protection Following Role Transitions

This is the most basic example in which a primary database uses high availability far sync instances to extend zero data loss failover to a remote standby database.

Ideally the high availability far sync instance is deployed in a location separate from the primary database to isolate it from site failure, but within a metro area distance (network RTT of 5ms or less – subject to performance testing). Even if no separate location is available there is still a benefit to deploying a far sync instance within the same data center to enable fast, zero data loss failover for all unrecoverable outages short of full site failure.

The remote high availability far sync instance is idle while the standby database is in a standby role. It becomes active when the standby database transitions to the primary database role, enabling zero data loss failover to the new standby (old primary). The high availability far sync instance that is local to the original primary database becomes inactive while it is in a standby role.

Figure 15-2 Role Transition Facilitated by Far Sync



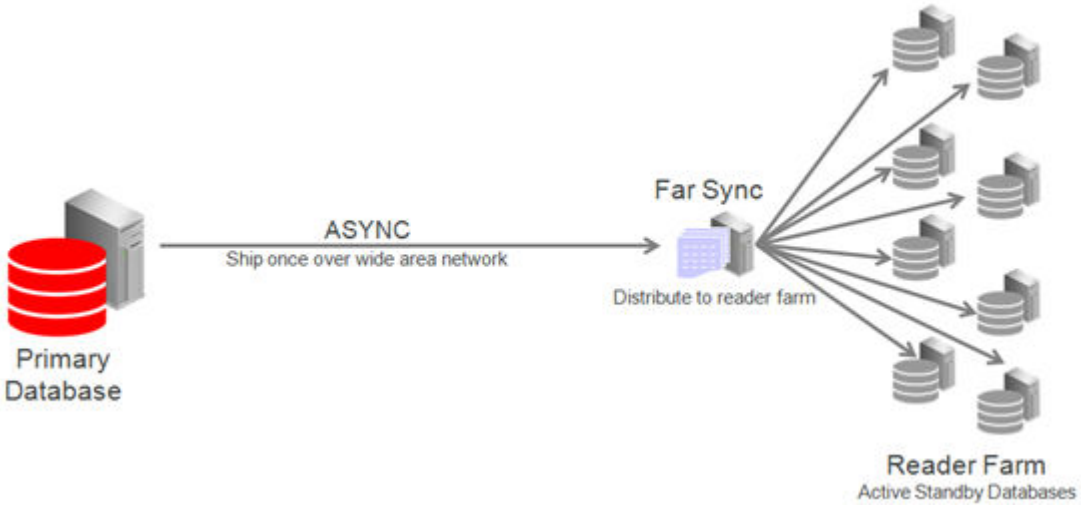
High availability far sync options are described in Far Sync Instance High Availability Typologies.

Case 2: Reader Farm Support

Far Sync can support up to 30 remote destinations, making it a very useful tool for supporting a reader farm – an Active Data Guard configuration having multiple active standby databases to easily scale read performance.

In this example the reader farm is configured in a remote location from the primary database. The primary ships once over the WAN to the far sync instance located in the remote destination and Far Sync distributes redo locally to all active standby databases in the reader farm.

Figure 15-3 Far Sync Ships Redo to Reader Farm



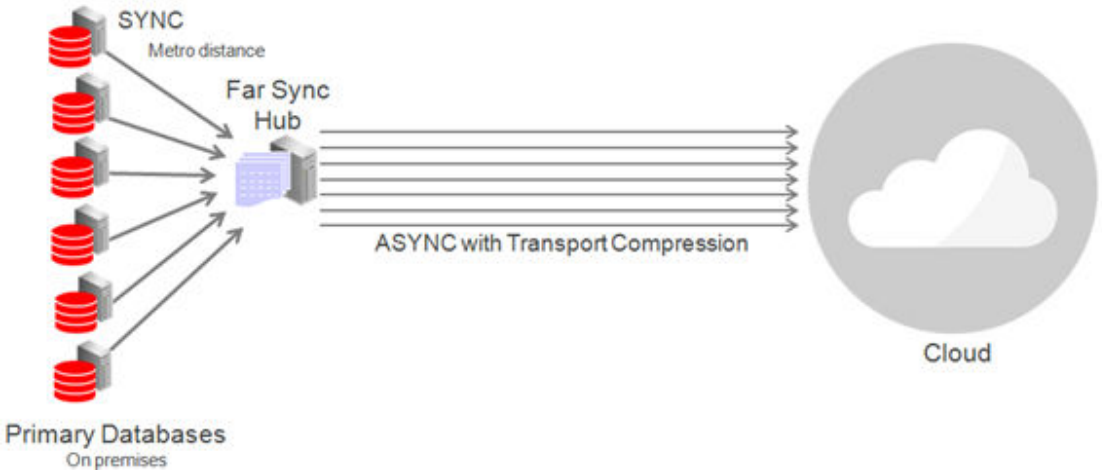
Case 3: Cloud Deployment With Far Sync Hub

Far Sync is a very lightweight process; a single physical server can support multiple far sync instances, each providing zero data loss failover to a remote destination.

The diagram below shows primary databases shipping to a single physical machine operating as a far sync "hub" consisting of multiple far sync instances on a single physical machine. Primaries and the far sync hub are on-premises while standby databases are deployed remotely in the cloud.

Note that all of the systems in this configuration (primary and standby database hosts and far sync instance host) must meet the usual requirements for compatibility in a Data Guard configuration described in Data Guard Support for Heterogeneous Primary and Physical Standbys in Same Data Guard Configuration (Doc ID 413484.1).

Figure 15-4 Far Sync Hub Architecture



Far Sync High Availability Topologies

To keep far sync instances highly available, consider the following deployment topologies.

Deploy Far Sync Instances on Oracle Real Application Clusters

The far sync instance can be placed on an Oracle RAC cluster. In this configuration a far sync instance is only active on one server at a time while other servers provide automatic failover for high availability. The characteristics of this approach include:

- Lowest data loss potential and brown-out when the active far sync instance or node fails.
- The ability to resume zero data loss protection quickly after far sync instance failure.
- By itself, this solution does not address cluster failure.

The most critical applications are well served by a pair of Oracle RAC far sync instances, each configured as an alternate for the other and deployed at different locations. This provides the most robust HA and data protection (during instance, node, cluster and site outages).

Deploy Far Sync Instances on Alternate Destinations and Multiple Far Sync instances

Configuring two separate far sync instances on distinct physical machines, each serving as an alternate destination for the other, provides far sync instance high availability in a non-Oracle RAC environment. Each destination defined on the primary database contains the ALTERNATE keyword assigning the other far sync instance as the alternate. When the active far sync instance enters an error state the alternate destination pointing to the alternate far sync instance is enabled automatically. By defining a far sync instance as an alternate destination, Maximum Availability protection will be maintained after a briefly dropping to a resynchronization state while the new destination is prepared.

The characteristics of this approach include:

- Retains zero data loss coverage after far sync instance transport failures (instance or network outages).
- Failure testing has shown
  - During far sync instance failures a performance brownout of approximately 3.5 seconds while SYNC redo transport starts (network sync service - NSS).

  - During network failures a short brownout equal to the setting of the destination's net_timeout parameter was observed.
- HA for machine outage assuming each far sync instance is on separate hardware.
- HA for site outage assuming far sync instances are deployed in separate sites.
- Higher application brown-out and resynchronization time during far sync instance outages compared with Far Sync with Oracle RAC

Deploy a Far Sync Instance on the Terminal Standby as an Alternate Destination

The simplest approach to maintaining data protection during a far sync instance outage is to create an alternate `LOG_ARCHIVE_DEST_n` pointing directly to the terminal standby (the terminal failover target). Asynchronous transport to the remote destination is the most likely choice in order to avoid the performance impact on the primary caused by WAN network latency.

Asynchronous transport can achieve near-zero data loss protection (as little as sub-seconds to seconds of exposure), but because it never waits for standby acknowledgment, it is unable to provide a zero data loss guarantee. In this configuration the protection level must be dropped to Maximum Performance prior to a switchover (planned event) as the level must be enforceable on the target in order to perform the transition. Changing protection levels and transport methods is a dynamic operation that does not require downtime.

During a far sync instance outage, redo transport automatically fails over to using the alternate destination. Once the far sync instance is repaired and resumes operation, transport automatically switches back to the far sync instance and zero data loss protection is restored.

The characteristics of this approach include:

- No additional hardware or far sync instances to manage.
- Loss of zero data loss coverage during a far sync instance outage. Data protection level drops to `UNSYNCHRONIZED` with `ASYNC` transport until the Far sync instance can resume operation and the standby become fully synchronized.

Choosing a Far Sync Deployment Topology

All configurations for far sync instance high availability perform equally with regard to receiving and sending redo. The choice of configuration should be based on application tolerance to the maximum data loss (RPO) and application brownout period of the different failure scenarios.

- Far sync instances deployed on Oracle RAC provides the lowest brownout and best protection however has no coverage for cluster or site outage. The most critical applications are well served by a pair of Oracle RAC far sync instances configured as alternates for each other and deployed at different locations. This provides the most robust Far Sync high availability (instance, node, cluster, and site failure) protection.
- Alternate far sync instances in a non-RAC environment provide the ability to place each instance on separate physical database servers. This configuration provides protection by deploying the far sync instances in different sites. Applications where data protection is critical but where cost is an important consideration are best served by deploying a pair of single node far sync instances, each as an alternate for the other. There is, however, slightly increased application brownout and longer resynchronization time while transport transitions from one far sync instance to the other. There is also the potential for data loss should a second outage impact the primary database while transport transitions from one far sync instance to the other.
- Terminal standby alternate configurations require that the application accept that there is no zero data loss protection while the far sync instance is not available, but requires no additional hardware to implement. Applications that can tolerate increased data loss

potential during a far sync instance outage, and where low cost is the main consideration, are best served by configuring the terminal standby as an alternate location using asynchronous redo transport. Use of the terminal standby as an alternate destination requires accepting that the configuration will run in asynchronous mode during the entire period required to resolve the far sync instance outage. The advantage of this approach is that it requires no additional hardware or software to deploy or manage. Applications that can tolerate increased data loss potential during a far sync instance outage and where low cost is the main consideration are best served by configuring the terminal standby as an alternate location using ASYNC redo transport.

- A Far Sync hub is an efficient way of consolidating far sync instances for multiple Data Guard configurations on a single physical host. Cloud deployments that include a zero data loss service level category can deploy a Far Sync hub to efficiently consolidate far sync instances for multiple zero data loss configuration on a single physical machine or cluster
- Applications where data protection is critical but where cost is an important consideration are best served by deploying a pair of single node far sync instances, each as an alternate for the other.

Far Sync Configuration Best Practices

The following are far sync configuration best practices that are necessary in addition to those best practices that apply to any synchronous redo transport destination.

- The network between the primary database and the far sync instance must:
  - Have round trip latency low enough so that the impact to response time and throughput of the primary database does not exceed business requirements. The degree of impact is very application specific and will require testing to validate. In general, experience shows that there is a higher likelihood of success if the round-trip latency is less than 5ms, though there are successful deployments at higher latencies.
  - Provide enough bandwidth between the primary database and the far sync instance to accommodate peak redo volumes, in addition to any other traffic sharing the network. Redo transport compression can be used to reduce network bandwidth requirements.
  - Ideally have redundant network links that also tolerate network component failure.
- Standard Oracle Data Guard network best practices, such as setting appropriate TCP send and receive buffer sizes equal to three times the bandwidth delay product. See Configure Online Redo Logs Appropriately.
- Standby redo logs for a far sync instance should be placed on storage with sufficient IOPS (writes per second) capacity to exceed the I/O of the LGWR process on the primary database during peak activity, in addition to any IOPS from other activities. This is an important consideration. For example:
  - If the far sync instance has lower performing disks than the primary database, it will not be able to forward redo to remote destinations as fast as it is received, and an archive log gap may form.
  - In redo gap resolution scenarios, due to planned maintenance on the standby or network outages, for example, there will be additional I/O requests for gap resolution on top of peak redo coming in.
  - Lower performing disks at the far sync instance will delay acknowledgment to the primary database, increasing the total round-trip time between primary and standby databases and impacting application response time. This impact can be eliminated by using Fast Sync between the primary database and the far sync instance.

- The far sync instance should follow the same standby redo log best practices as the standby database. See Configure Online Redo Logs Appropriately.
- The standby redo logs of an alternate far sync instance should be manually cleared before use to achieve the fastest return to synchronous transport when the alternate far sync is activated. For example:

```
ALTER DATABASE CLEAR LOGFILE GROUP 4, GROUP 5, GROUP 6;
```

- Oracle MAA performance testing shows that a small far sync instance SGA does not impact the performance of the far sync instance or the primary database. To conserve system resources, you can configure the minimum SGA required for Far Sync to function.
  - Set `CPU_COUNT=4`. Values of 1 or 2 are possible when neither compression nor encryption are not being used.
  - Reducing the `CPU_COUNT` during testing has no effect on the performance of the Far sync instance.
- Configure far sync instances for both the primary and standby databases to maintain zero data loss protection following role transitions. The second far sync instance configured in proximity to the standby database is idle until the standby becomes the primary database, enabling synchronous redo transport in the reverse direction.

  Note that in a Data Guard Broker configuration, a switchover (planned role transition) cannot occur while in Maximum Availability mode unless the protection mode can be enforced from the target standby site. If the standby database does not have its own far sync instance it will have to be configured to ship asynchronous redo to the original primary database after the roles are reversed. This prevents a switchover from occurring unless the protection mode for the primary database is first dropped from Maximum Availability to Maximum Performance.
- Fast Sync yields a 4% to 12% primary database performance improvement compared to synchronous transport, depending on the network latency and the I/O speed of the far sync instance hardware.
- Provided CPU, I/O, and network requirements are met.
  - Placing the far sync instance on a virtual machine produces no reduction in performance over physical hardware configurations.
  - Multiple far sync instances servicing multiple Data Guard configurations can share the same physical server, cluster, or virtual machine.
- Note that archives may need to be managed on the far sync server.

Configuring the Active Data Guard Far Sync Architecture

The following topics walk you through an example of configuring an Active Data Guard Far Sync architecture.

Configuring the Far Sync Instances

The following examples show you how to add far sync instances to an Oracle Data Guard broker configuration.

The first step is to add a far sync standby instance that is independent or fault isolated from the primary database server, and where the network latency between the primary server and the far sync server is consistently low enough that application performance can tolerate it (for example, < 5 ms).

In the following example, far sync instance FS1 is created for the primary database, North_Sales.

```
DGMGRL> ADD FAR_SYNC FS1 AS CONNECT IDENTIFIER IS FS1.example.com;
Far Sync FS1 added
DGMGRL> ENABLE FAR_SYNC FS1;
Enabled.
DGMGRL> SHOW CONFIGURATION;

Configuration - DRSolution

  Protection Mode: MaxPerformance
  Members:
  North_Sales  - Primary database
    FS1        - Far Sync
    South_Sales - Physical standby database

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS
```

After a far sync instance has been added to the configuration, set up redo transport to support maximum availability mode and then upgrade the protection mode, as shown in the following example.

```
DGMGRL> EDIT DATABASE 'North_Sales' SET PROPERTY 'RedoRoutes' = '(LOCAL : FS1
SYNC)';
DGMGRL> EDIT FAR_SYNC 'FS1' SET PROPERTY 'RedoRoutes' = '(North_Sales :
South_Sales ASYNC)';
DGMGRL> EDIT CONFIGURATION SET PROTECTION MODE AS MaxAvailability;
DGMGRL> SHOW CONFIGURATION;

Configuration - DRSolution

  Protection Mode: MaxAvailability
  Members:
  North_Sales  - Primary database
    FS1          - Far Sync
      South_Sales - Physical standby database

Fast-Start Failover: DISABLED
Configuration Status:
SUCCESS
```

To ensure that maximum availability protection mode can be maintained when the remote standby database, South_Sales, becomes the primary database after a switchover or a failover, add a second far sync instance to the configuration so that South_Sales can send redo in synchronous mode, which in turn will send redo to the new terminal database, North_Sales, after the role transition.

The following example shows you how to add a second far sync instance (FS2) to the broker configuration.

```
DGMGRL> ADD FAR_SYNC FS2 AS CONNECT IDENTIFIER IS FS2.example.com;
Far Sync FS2 added
DGMGRL> EDIT FAR_SYNC 'FS2' SET PROPERTY 'RedoRoutes' = '(South_Sales :
North_Sales ASYNC)';
DGMGRL> ENABLE FAR_SYNC FS2;
Enabled.
DGMGRL> EDIT DATABASE 'South_Sales' SET PROPERTY 'RedoRoutes' = '(LOCAL : FS2
SYNC)';
DGMGRL> SHOW CONFIGURATION;

Configuration - DRSolution

  Protection Mode: MaxAvailability
  Members:
  North_Sales  - Primary database
    FS1          - Far Sync
      South_Sales - Physical standby database
      FS2         - Far Sync (inactive)

Fast-Start Failover: DISABLED
Configuration Status:
SUCCESS
```

Setting Up HA Far Sync Instances

Alternate HA far sync instances are set up to provide high availability for the far sync instances you created for the primary and remote standby databases.

The following example shows you how to add a second far sync instance (FS1a) to the primary database's far sync instance (FS1) in the Oracle Data Guard broker configuration, so that if the primary far sync instance becomes unavailable, redo transport will use the alternate far sync instance.

```
DGMGRL> ADD FAR_SYNC FS1a AS CONNECT IDENTIFIER IS FS1a.example.com;
Far Sync FS1a added
DGMGRL> EDIT DATABASE 'North_Sales' SET PROPERTY 'RedoRoutes' = ' (LOCAL:(FS1
SYNC PRIORITY=1, FS1a SYNC PRIORITY=2))';
DGMGRL> EDIT FAR_SYNC 'FS1' SET PROPERTY 'RedoRoutes' = '(North_Sales :
South_Sales ASYNC)';
DGMGRL> EDIT FAR_SYNC 'FS1a' SET PROPERTY 'RedoRoutes' = '(North_Sales :
South_Sales ASYNC)';
DGMGRL> EDIT CONFIGURATION SET PROTECTION MODE AS MaxAvailability;
DGMGRL> SHOW CONFIGURATION;

Configuration - DRSolution

  Protection Mode: MaxAvailability
  Members:
  North_Sales  - Primary database
    FS1          - Far Sync
    FS1a          - Far Sync
      South_Sales - Physical standby database
```

```
Fast-Start Failover: DISABLED
Configuration Status:
SUCCESS
```

After adding the alternate far sync instance on the primary, use the following example to add an alternate far sync instance (FS2a) on the standby.

```
DGMGRL> ADD FAR_SYNC FS2a AS CONNECT IDENTIFIER IS FS2a.example.com;
Far Sync FS2a added
DGMGRL> EDIT DATABASE 'South_Sales' SET PROPERTY 'RedoRoutes' = ' (LOCAL:(FS2
SYNC PRIORITY=1, FS2a SYNC PRIORITY=2))';
DGMGRL> EDIT FAR_SYNC 'FS2' SET PROPERTY 'RedoRoutes' = '(South_Sales :
North_Sales ASYNC)';
DGMGRL> EDIT FAR_SYNC 'FS2a' SET PROPERTY 'RedoRoutes' = '(South_Sales :
North_Sales ASYNC)';
DGMGRL> EDIT CONFIGURATION SET PROTECTION MODE AS MaxAvailability;
DGMGRL> SHOW CONFIGURATION;

Configuration - DRSolution

  Protection Mode: MaxAvailability
  Members:
    North_Sales - Primary database
      FS1 - Far Sync
      FS1a - Far Sync
        South_Sales - Physical standby database
          FS2 - Far Sync (inactive)
          FS2a - Far Sync (inactive)

Fast-Start Failover: DISABLED
Configuration Status:
SUCCESS
```

Configuring Far Sync Instances with Oracle RAC or Oracle Clusterware

If a far sync instance is deployed on a server or cluster with Oracle Clusterware (for example, in an Oracle Restart, Oracle Real Application Clusters (Oracle RAC), or Oracle RAC One Node installation), then use the SRVCTL utility to specify a default open mode of mount.

You can use a command such as the following:

```
srvctl modify database -d db_unique_name -startoption MOUNT
```

Encrypting a Database Using Data Guard and Fast Offline Encryption

Encrypting a database using Transparent Data Encryption (TDE) can be done more quickly, with minimal down time, and no extra space requirements, by using the standby database and offline encryption.

In this two-phase process, the standby database is encrypted offline, followed by a switchover, and then the offline encryption is repeated on the new standby database (formerly the primary).

In more recent Oracle releases online encryption is also available. Online encryption may fit the needs for some, but requires additional storage while a tablespace is converted, and online encryption can be a time-consuming process because each block is read and written to a new encrypted data file. With fast offline encryption, each data file is encrypted directly, in-place, on a mounted standby database.

Step 1: Configure Transparent Data Encryption (TDE)

There are a number of different TDE configuration options. Different Oracle releases have different requirements. It is strongly recommended that you review Introduction to Transparent Data Encryption in the *Oracle Database Advanced Security Guide* for your database release to understand the configuration options and implications of TDE.

> **Note:**
>
> This process describes configuring a united, file-based keystore, which means that the wallets are stored on a file system, and all keys for all PDBs are stored in a single wallet.
> For more complex configurations such as isolated PDBs, Oracle Key Vault (OKV), or Hardware Security Module (HSM), see Using Transparent Data Encryption in *Oracle Database Advanced Security Guide* for details.

The following are the basic parameters required to configure a united, file-based keystore. The parameters are configured on the primary and standby databases but may have different values.

Parameter	Configuration Best Practice

`WALLET_ROOT`	Starting in Oracle Database 18c, configuring the `WALLET_ROOT` database parameter is the best practice for specifying the root directory for all database wallets. For clustered databases, the location specified in `WALLET_ROOT` must be a shared location such as an ASM disk.
	
	`ALTER SYSTEM SET`
	`WALLET_ROOT='+DATA/db_unique_name'`
	`SCOPE=SPFILE SID='*';`
	
	Note:
	`WALLET_ROOT` is a static parameter; the database must be restarted for the changes to take effect. `TDE_CONFIGURATION` cannot be set until the database is restarted with the `WALLET_ROOT` set.
	
	See WALLET_ROOT in the *Oracle Database Reference* for more details.

Parameter	Configuration Best Practice
TDE_CONFIGURATION	The TDE_CONFIGURATION database parameter sets the type of keystore. TDE_CONFIGURATION is dynamic, but it can only be set after the database is restarted with WALLET_ROOT configured. ALTER SYSTEM SET TDE_CONFIGURATION='KEYSTORE_CONFIGURATION=FILE' SCOPE=SPFILE SID='*'; See TDE_CONFIGURATION in the *Oracle Database Reference* for more details.
TABLESPACE_ENCRYPTION	The database parameter TABLESPACE_ENCRYPTION is available in Oracle 19c (19.16). TABLESPACE_ENCRYPTION is an alternative to ENCRYPT_NEW_TABLESPACES, which specifies whether to encrypt new tablespaces when they are created. If both TABLESPACE_ENCRYPTION and ENCRYPT_NEW_TABLESPACES parameters are set, TABLESPACE_ENCRYPTION takes precedence. The values for TABLESPACE_ENCRYPTION are as follows: • AUTO_ENABLE - All newly created tablespaces will be encrypted. This is the Oracle Cloud default which cannot be overridden in Oracle 19c (19.16) and later. • MANUAL_ENABLE - Manually control whether tablespaces are encrypted with the ENCRYPTION clause on the CREATE statement. • DECRYPT_ONLY - No tablespaces will be encrypted. This setting is used in a hybrid Data Guard configuration where the on-premises database remains unencrypted while the cloud database is encrypted. ALTER SYSTEM SET TABLESPACE_ENCRYPTION=MANUAL_ENABLE SCOPE=BOTH SID='*'; See TABLESPACE_ENCRYPTION in the *Oracle Database Reference* for more details.

Parameter	Configuration Best Practice
ENCRYPT_NEW_TABLESPACES	Before Oracle Database 19c (19.16) the ENCRYPT_NEW_TABLESPACES parameter specifies whether to encrypt new tablespaces. The values for ENCRYPT_NEW_TABLESPACES are as follows: • CLOUD_ONLY - When a tablespace is created in the Oracle Cloud, it is transparently encrypted with the default encryption algorithm, whether or not the encryption clause in included. The default encryption algorithm can be changed as shown in Step 2, but AES128 is the default algorithm. • ALWAYS - The new tablespace is transparently encrypted whether or not the database is in the Oracle Cloud. • DDL - Manually control whether tablespaces are encrypted with the ENCRYPTION clause. See ENCRYPT_NEW_TABLESPACES in the *Oracle Database Reference* for more details.

The following table indicates which TDE parameters to configure based on your Oracle Database release.

Oracle Release	WALLET_ROOT and TDE_CONFIGURATION	TABLESPACE_ENCRYPTION	ENCRYPT_NEW_TABLESPACES
Oracle 19c (19.16) and later	Yes	Yes	No
Oracle 18c to 19c (19.15)	Yes	No	Yes

Step 2: Set the default encryption algorithm

The default encryption algorithm for TDE is AES128. In Oracle 21c and later releases, the algorithm can be set with the TABLESPACE_ENCRYPTION_DEFAULT_ALGORITHM parameter, but this setting must be configured **before** creating the wallet. Likewise, "_tablespace_encryption_default_algorithm" can be used in Oracle 19c and earlier with patch 30398099.

This setting determines the encryption algorithm used on new tablespaces for TABLESPACE_ENCRYPTION=AUTO_ENABLE, ENCRYPT_NEW_TABLESPACES=ALWAYS, and for offline encryption used in this process.

On the primary and standby databases issue:

```
-- for Oracle 21c and later
ALTER SYSTEM SET "tablespace_encryption_default_algorithm"='AES256'
scope=both;

 -- for Oracle 19c and earlier
ALTER SYSTEM SET "_tablespace_encryption_default_algorithm"='AES256'
scope=both;
```

Step 3: Create the encryption wallet and set the master key

The TDE documentation is very thorough in describing creation of the wallet, or keystore, and setting the master encryption key on the primary database.

See Configuring a Software Keystore and TDE Master Encryption Key for United Mode in *Oracle Database Advanced Security Guide* for details.

Note that even if the primary database is to remain unencrypted after the standby is encrypted, in a hybrid Data Guard use case the master key **must** be set on the primary database. This key is used to encrypt data on the standby during redo apply, and after role transition. The key is be used to decrypt data from the encrypted primary cloud database after role transition.

Step 4: Copy the wallet files to the standby database environment

The standby database must have a copy of the encryption wallet and the auto-login keystore to perform encryption operations on the standby database. Copy the files from the primary database to the standby database accordingly.

From the location defined by `WALLET_ROOT`. If the target directory does not exist on the standby, it should be created manually.

Copy files to each node:

```
ASMCMD> cp +DATA/PRIMARY_ORACLE_UNQNAME/TDE/cwallet.sso /tmp
ASMCMD> cp +DATA/PRIMARY_ORACLE_UNQNAME/TDE/ewallet.p12 /tmp

<primary host>$ scp /tmp/cwallet.sso ewallet.p12 oracle@standby_host:/tmp

<standby host> ASMCMD> cp /tmp/cwallet.sso +DATA/STANDBY_db_unique_name/TDE/
<standby host> ASMCMD> cp /tmp/ewallet.p12 +DATA/STANDBY_db_unique_name/TDE/
```

Alternatively, the files can be copied directly from ASM to ASM.

```
ASMCMD>cp cwallet.sso sys/password@stbyhost1.+ASM1:+DATA/
STANDBY_ORACLE_UNQNAME/TDE/
ASMCMD>cp ewallet.p12 sys/password@stbyhost1.+ASM1:+DATA/
STANDBY_ORACLE_UNQNAME/TDE/
```

Step 5: Verify Data Guard health

Before starting the offline encryption process, make sure that the standby database is current with the primary. Managed recovery must be stopped during the encryption process, and so ensuring that the standby database is current with the primary reduces the redo gap that must be applied after the encryption process.

On the primary or standby database, look up redo apply lag, then validate the standby database as shown in the following example. The Data Guard Broker command `VALIDATE DATABASE` lists potential configuration gaps. Address any gaps and verify that the status of "Ready for Switchover" and "Ready for Failover" are both `YES`.

```
DGMGRL> SHOW CONFIGURATION LAG

Configuration - dgconfig

  Protection Mode: MaxPerformance
  Members:
primary_db    - Primary database
  standby_db - Physical standby database
               Transport Lag:       0 seconds (computed 1 second ago)
               Apply Lag:           0 seconds (computed 1 second ago)
```

```
Fast-Start Failover:  Disabled

Configuration Status:
SUCCESS   (status updated 11 seconds ago)


DGMGRL> VALIDATE DATABASE <standby>

  Database Role:     Physical standby database
  Primary Database:  primary

  Ready for Switchover:  Yes
  Ready for Failover:    Yes (Primary Running)
```

Step 6: Place the standby database in a mounted state with recovery stopped

Before you run the offline encryption process directly against the data files, the standby database must be mounted and recovery must be stopped. All instances of the standby can be used during the encryption process to encrypt multiple files simultaneously.

```
$ srvctl stop database -d standby -o immediate

$ srvctl start database -d standby -o mount

DGMGRL> EDIT DATABASE standby SET STATE=APPLY-OFF;
```

Redo transport services continue to ship redo to ensure that the archived logs are present at the standby database. This process maintains Recovery Point Objective (RPO) in the event of a failure during the encryption process.

For a database that is very active, the required number of archived logs could be significant, so make sure that there is sufficient space in the recovery area.

Step 7: Encrypt data files in-place and in parallel on the standby database

The encryption properties of TEMP tablespaces cannot be changed after creation. To encrypt a TEMP tablespace it must be created as encrypted.

To use an encrypted TEMP tablespace, create a new TEMP tablespace using the `ENCRYPTION` clause and make it the default temporary tablespace. Then drop the original TEMP tablespace.

```
SQL> CREATE TEMPORARY TABLESPACE TEMP_ENC ENCRYPTION ENCRYPT;

SQL> ALTER DATABASE DEFAULT TEMPORARY TABLESPACE TEMP_ENC;
```

The UNDO and TEMP metadata that is generated from sensitive data in an encrypted tablespace is already encrypted; therefore, encrypting UNDO and TEMP tablespaces is optional.

1. Make sure the standby database is mounted and the keystore is open.

   ```
   SQL> select inst_id,database_role,open_mode from gv$database;

      INST_ID DATABASE_ROLE    OPEN_MODE
   ---------- ---------------- --------------------
            1 PHYSICAL STANDBY MOUNTED
   ```

```
         2 PHYSICAL STANDBY MOUNTED

SQL> col WRL_PARAMETER format a40
SQL> set linesize 120 pagesize 9999
SQL> select * from gv$encryption_wallet;

INST_ID WRL_TYPE WRL_PARAMETER                            STATUS
------- -------- ---------------------------------------- ------
      1 file     +DATA/ORACLE_UNQNAME/TDE                 OPEN
```

2. Encrypt the data files.

   The offline encryption command encrypts each data file with a single process; however, multiple data files can be encrypted in parallel with separate sessions. Each session can fully utilize a CPU core. It is recommended that each instance issues a number of sessions less than or equal to the number of cores on the host.

   The following query can be used to generate a script to convert the data files. Break the script into multiple scripts and run each smaller script in an individual session. The most efficient process is to encrypt large files individually while placing multiple smaller files in a separate script.

   > **Note:**
   >
   > The seed database files do not need to be encrypted.

```
set lines 120
set pages 9999
spool encrypt.sql
select 'alter session set container='||pdb.name||';'||chr(10)||'alter
database datafile '||chr(39)||df.name||chr(39)||' encrypt;' COMMAND
from v$tablespace ts, v$datafile df, v$pdbs pdb where ts.ts#=df.ts# and
ts.con_id=df.con_id and df.con_id=pdb.con_id and pdb.name <> 'PDB$SEED';

spool off
COMMAND
--------------------------------------------------------------------------------
-------------------------------------------
alter session set container=ORADBP11;
alter database datafile '+DATA/DB_UNIQUE_NAME/
E73F249E7030C3B8E0537B544664A065/DATAFILE/system.336.1113852973' encrypt;

alter session set container=ORADBP11;
alter database datafile '+DATA/DB_UNIQUE_NAME/
E73F249E7030C3B8E0537B544664A065/DATAFILE/sysaux.335.1113852973' encrypt;

alter session set container=ORADBP11;
alter database datafile '+DATA/DB_UNIQUE_NAME/
E73F249E7030C3B8E0537B544664A065/DATAFILE/undotbs1.337.1113852973' encrypt;

<...>
```

3. TEMP files can be encrypted by dropping and recreating them using the `ENCRYPTION` clause in the `CREATE` statement. Identify existing TEMP files using the `V$TEMPFILE` view.

4. Validate that all data files are encrypted by querying `V$DATAFILE_HEADER.ENCRYPTED`. After file encryption is completed, the `ENCRYPTED` column indicates whether the file is encrypted (`YES`) or not (`NO`). All data files except those belonging to the seed PDB should be encrypted.

Step 8: Restart redo apply and catch up on the standby database

After it is confirmed that all data files are encrypted, the standby database must apply all of the redo from the primary that was generated during the encryption process. The following are recommended ways to catch up redo on the standby database, depending on the amount of redo that needs to be applied.

- If the gap is small, restart managed recovery and apply the redo gap until the apply lag is 0.

  On the primary or standby database run

  ```
  DGMGRL> edit database standby set state=apply-on;
  ```

- If the encryption process took longer, and the primary database was very active, the gap might be large. It is often faster to use an incremental roll forward approach to copy only the blocks which have changed since apply was stopped.

  That process is described in My Oracle Support note How to Roll Forward a Standby Database Using Recover Database From Service (Doc ID 2850185.1). Recovery is still needed when the roll forward is complete, but this process can shorten the time significantly to close large gaps.

Step 9: Perform a Data Guard switchover to begin encryption on the primary database

Until you are ready to encrypt the primary database, you can allow the unencrypted primary database to ship unencrypted redo to the standby, where it is encrypted by the standby indefinitely.

When you are ready to encrypt the primary database, and it is convenient to switch the database roles, perform a Data Guard switchover, making the encrypted standby database the new primary and the unencrypted primary database the new standby.

On the original primary database which is now the standby, repeat steps 5-8 to encrypt the data files and catch up on redo.

Step 11: Perform a Data Guard switchover (optional)

If, after both the standby and primary database are encrypted, you prefer to revert to the original primary-standby database roles, you can perform a Data Guard switchover to re-establish their original roles.

16

Tune and Troubleshoot Oracle Data Guard

When redo transport, redo apply, or role transitions are not meeting your expected requirements, use the following guidelines to help you tune and troubleshoot your deployment.

Overview of Oracle Data Guard Tuning and Troubleshooting

To get the best performance from your Oracle Data Guard configuration, use the following Oracle MAA best practices for monitoring, assessment, and performance tuning.

- Ensure that Oracle Database and Oracle Data Guard configuration best practices are in place.

  The assumption when assessing and tuning is that all of the Oracle Database and Data Guard configuration best practices are already integrated in the environment. Evaluate the adherence to those best practices before doing any tuning.

- Assess and tune redo transport services

  Oracle Data Guard automatically tunes redo transport to optimize performance. However, if you observe performance issues, you can monitor and tune redo transport services.

  Asynchronous redo transport with Maximum Performance data protection mode is the default Oracle Data Guard configuration. Tuning asynchronous redo transport consists mainly of ensuring that the primary, standby, and network resources are sufficient for handling the workload, and that you monitor those resources for bottlenecks.

  Synchronous redo transport does sacrifice some performance for zero data loss; however, using sound MAA recommended methods, you can monitor and assess the impact and distribute resources appropriately.

- Assess and tune redo apply

  In most cases, the default Oracle settings result in satisfactory performance for media recovery when the standby is always up to date. However, as applications and databases increase in size and throughput, media recovery operations can benefit from additional tuning to further optimize recovery time or redo apply throughput on a standby database

- Assess and tune role transitions

  With proper planning and implementation, Oracle Data Guard and Active Data Guard role transitions can effectively minimize downtime and ensure that the database environment is restored with minimal impact on the business. Performance tests using a physical standby database and Oracle Maximum Availability Architecture (MAA) best practices have shown that switchover and failover can be reduced to seconds.

Redo Transport Troubleshooting and Tuning

Oracle Data Guard redo transport performance is directly dependent on the performance of the primary and standby systems, the network that connects them, and the I/O subsystem.

For most Oracle Data Guard configurations, you should be able to achieve zero or minimal data loss by troubleshooting and tuning redo transport.

The guidance presented here assumes that the MAA configuration best practices are followed. As a prerequisite, ensure that the Oracle Data Guard Configuration Best Practices are implemented.

To improve transport holistically, leverage the data gathering and troubleshooting methodology described in the topics below, which guide you through gathering the necessary data to assess whether there is indeed a redo transport problem and what can be tuned to optimize redo transport throughput.

Gather Topology Information

Understanding the topology of the Oracle Data Guard configuration, and its relevance to Data Guard performance, helps eliminate infrastructure weaknesses that are often incorrectly attributed to the Data Guard architecture.

Oracle recommends that you outline the following high-level architecture information.

- Describe the primary and standby database system (number of nodes in Oracle RAC cluster, CPUs and memory per database node, storage I/O system)
- Describe network topology connecting the primary and standby systems
  - Network components/devices in between primary and standby
  - Network bandwidth and latency

For standby databases with symmetric hardware and configuration, and with a good tuned network configuration, the transport lag should be less than 10 seconds and in most cases less than 1 second.

Verify Transport Lag and Understand Redo Transport Configuration

To determine if there is any lag on the standby database, and if this is a transport or apply lag, query the `V$DATAGUARD_STATS` view.

```
SQL> select name,value,time_computed,datum_time from v$dataguard_stats where
name='%lag';
```

The `DATUM_TIME` column is the local time on the standby database when the datum used to compute the metric was received. The lag metrics are computed based on data that is periodically received from the primary database. An unchanging value in this column across multiple queries indicates that the standby database is not receiving data from the primary database. The potential data loss in this scenario would be from the last datum time from `V$DATAGUARD_STATS` to the current time on the standby.

To obtain a histogram that shows the history of transport or apply lag values since the standby instance was last started, query the V$STANDBY_EVENT_HISTOGRAM view.

```
SQL> select * from v$standby_event_histogram where name like '%lag' and count >0;
```

To evaluate the transport or apply lag over a time period, take a snapshot of V$STANDBY_EVENT_HISTOGRAM at the beginning of the time period and compare that snapshot with one taken at the end of the time period.

```
SQL> col NAME format a10
SQL> select NAME,TIME,UNIT,COUNT,LAST_TIME_UPDATED from
V$STANDBY_EVENT_HISTOGRAM where
 name like '%lag' and count >0 order by LAST_TIME_UPDATED;

NAME             TIME UNIT                  COUNT LAST_TIME_UPDATED

---------- ---------- ---------------- ---------- --------------------

transport lag      41 seconds                   3      01/05/2022 16:30:59

transport lag     245 seconds                   1      01/05/2022 16:31:02

transport lag     365 seconds                   2      01/05/2022 16:31:03

transport lag     451 seconds                   2      01/05/2022 16:31:04
```

If you observe a high redo transport lag, continue this redo transport investigation with Gather Information to Troubleshoot Transport Lag. If you see no transport lag but a high redo apply lag, address the apply lag using the methodology in Redo Apply Troubleshooting and Tuning.

Gather Information to Troubleshoot Transport Lag

Gather the following information and investigate the questions when an unacceptable redo transport lag is observed:

- When did the transport lag occur? Record the V$DATAGUARD_STATS and V$STANDBY_EVENT_HISTOGRAM data to show when the lag started and how the lag is changing over time.
- Does the transport lag occur during certain time period, such as daily at 12 midnight for daily batch operations, monthly during large batch operation, or quarterly during quarter end?
- Check the LOG_ARCHIVE_DEST setting for any enabled Oracle Data Guard transport, and verify whether redo COMPRESSION or ENCRYPTION is enabled. Overall redo transport throughput can be negatively impacted because redo must be compressed or encrypted before sending, and then uncompressed or unencrypted upon receiving it on the standby. Verify if that change was recent, and if you can test disabling these setting attributes.
- Check the Oracle Net settings to evaluate if Oracle Net encryption is enabled. If Oracle Net encryption is enabled, when was it enabled and at what level? Oracle Net encryption can slow down redo throughput significantly because redo is encrypted before sending and unencrypted upon receiving the redo on the standby. Optionally, disable or reduce encryption levels to see if the redo transport lag reduces.

Compare Redo Generation Rate History on the Primary

There are cases where the primary database redo generation rate is exceptionally high for a short period of time, such as during large batch jobs, data loads, data pump operations, create table as select, PDML operations, or end of month, quarter, or year batch updates.

Obtain the redo generation history from the primary database and compare that to when the redo transport or redo apply lag started. Check if the redo generation rate is exceptionally high because of additional workloads, such as adding new pluggable databases or new application services. By doing so, additional tuning may be required to accommodate this additional load.

As part of troubleshooting, gather the following information or address the following questions:

- Gather daily history of primary database's redo generation rate using this query.

```
SQL> select trunc(completion_time) as "DATE", count(*) as "LOG SWITCHES",
round(sum(blocks*block_size)/1024/1024) as "REDO PER DAY (MB)"
from v$archived_log
where dest_id=1
group by trunc(completion_time) order by 1;
```

- Gather per log redo generation rate starting 6 hours prior to start any redo or transport lag.

```
SQL> alter session set nls_date_format='YYYY/MM/DD HH24:MI:SS';
SQL> select thread#,sequence#,blocks*block_size/1024/1024 MB,(next_time-
first_time)*86400 sec, blocks*block_size/1024/1024)/((next_time-
first_time)*86400) "MB/s" from v$archived_log
where ((next_time-first_time)*86400<>0)
and first_time between to_date('2015/01/15 08:00:00','YYYY/MM/DD
HH24:MI:SS')
and to_date('2015/01/15 11:00:00','YYYY/MM/DD HH24:MI:SS')
and dest_id=1 order by first_time;
```

- Gather hourly snapshots of the redo generation rate from the Automatic Workload Repository (AWR) report 6 hours before the start of any redo or transport lag.

  By default, Oracle Database automatically generates snapshots once every hour; however, you may want to manually create snapshots to capture statistics at times different from those of the automatically generated snapshots. To view information about an existing snapshot, use the `DBA_HIST_SNAPSHOT` view.

  See Creating Snapshots in the *Oracle Database Performance Tuning Guide* for complete information about AWR and generating snapshots and AWR reports.
- Is this primary redo generation rate exceptionally high compared to prior history?
- If possible, determine the workload that corresponds to the high redo generation rate and evaluate whether it's transient or if it can be tuned.

  For example, for large purge operations, consider truncate or drop partition operations to reduce the redo generation volumes.

Evaluate the Transport Network and Tune

Redo transport consists of the primary database instance background process sending redo to the standby database background process. You can evaluate if the network is optimized for Oracle Data Guard redo transport.

If asynchronous redo transport is configured, redo data is streamed to the standby in large packets asynchronously. To tune asynchronous redo transport over the network, you need to optimize a single process network transfer.

If synchronous redo transport is configured, each redo write must be acknowledged by the primary and standby databases before proceeding to the next redo write. You can optimize standby synchronous transport by using the `FASTSYNC` attribute as part of the `LOG_ARCHIVE_DEST` setting, but higher network latency (for example > 5 ms) impacts overall redo transport throughput.

Before you continue, see Assessing and Optimizing Network Performance first to:

- Assess whether you have sufficient network bandwidth to support the primary's redo generation rate
- Determine optimal TCP socket buffer sizes to tune redo transport
- Tune operating system limits on socket buffer sizes to tune redo transport
- Determine optimal MTU setting for redo write size
- Tune MTU to increase network throughput for redo transport

If network configuration is tuned, evaluate if the transport lag (refer to Verify Transport Lag and Understand Redo Transport Configuration) is reducing to acceptable levels. If that's the case, you have met your goals and you can stop. Otherwise continue with the rest of the rest of tuning and troubleshooting section.

Gather and Monitor System Resources

Gather Oracle Linux OSwatcher or Oracle Exadata Exawatcher data to analyze system resources.

OSWatcher (oswbb) is a collection of UNIX shell scripts intended to collect and archive operating system and network metrics to aid support in diagnosing performance issues. As a best practice, you should install and run OSWatcher on every node that has a running Oracle instance. In the case of a performance issue, Oracle support can use this data to help diagnose performance problems which may outside the database.

You can download OSWatcher from OSWatcher (Doc ID 301137.1).

ExaWatcher is a utility that collects performance data on the storage servers and database servers on an Exadata system. The data collected includes operating system statistics, such as iostat, cell statistics (cellsrvstat), and network statistics.

See Using ExaWatcher Charts to Monitor Exadata Database Machine Performance in the *Oracle Exadata Database Machine Maintenance Guide* for more information.

Tune to Meet Data Guard Resource Requirements

Redo transport can be impacted if:

- Primary or standby database is completely CPU bound
- Primary or standby database I/O system is saturated
- Network topology can't support the redo generation rates

Evaluate whether the primary database system has:

- Sufficient CPU utilization for Log Writer Process (LGWR) to post foregrounds efficiently
- Sufficient I/O bandwidth so local log writes maintain low I/O latency during peak rates

- Network interfaces that can handle peak redo rate volumes combined with any other network activity across the same interface
- Automatic Workload Repository (AWR), Active Session History (ASH), and OSwatcher or Exawatcher data gathered from the primary database for tuning and troubleshooting

Evaluate whether the standby database system has:

- Sufficient CPU utilization for the remote file server (RFS), the Oracle Data Guard process that receives redo at the standby database, to efficiently write to standby redo logs
- Sufficient I/O bandwidth to enable local log writes to maintain low I/O latency during peak rates
- A network interface that can receive the peak redo rate volumes combined with any other network activity across the same interface
- AWR, ASH, and OSwatcher or Exawatcher data gathered from the standby database for tuning and troubleshooting

> **Note:**
>
> The top issue encountered with the standby database is poor standby log write latency because of insufficient I/O bandwidth. This problem can be mitigated by using Data Guard Fast Sync.

If system configuration is tuned and the above resource constraints are removed, evaluate if the transport lag (refer to Verify Transport Lag and Understand Redo Transport Configuration) is reducing to acceptable levels. If that's the case, you have met your goals.

Advanced Troubleshooting: Determining Network Time with Asynchronous Redo Transport

Before you proceed, first see Assessing and Optimizing Network Performance.

Given enough resources, especially network bandwidth, asynchronous redo transport can maintain pace with very high workloads. In cases where resources are constrained, asynchronous redo transport can begin to fall behind resulting in a growing transport lag on the standby database.

Asynchronous redo transport (ASYNC) transmits redo data asynchronously with respect to transaction commitment. A transaction can commit without waiting for an acknowledgment that the redo generated by that transaction was successfully transmitted to a remote standby database. With ASYNC, the primary database Log Writer Process (LGWR) continues to acknowledge commit success even if limited bandwidth prevents the redo of previous transactions from being sent to the standby database immediately (picture a sink filling with water faster than it can drain).

ASYNC uses a TT00 process to transmit redo directly from the log buffer of the primary database. If the TT00 process is unable to keep pace, and the log buffer is recycled before the redo can be transmitted to the standby database, then the TT00 process automatically transitions to reading and sending from the online redo log file (ORL) on disk. Once TT00 transmission has caught up with current redo generation, it automatically transitions back to reading and sending directly from the log buffer.

In cases in which there are two or more log switches before the TT00 has completed sending the original ORL, the TT00 will still transition back to reading the contents of the current online log file. Any ORLs that were archived in between the original ORL and the current ORL are automatically transmitted using Oracle Data Guard's redo gap resolution process.

Sufficient resources, such as network bandwidth, CPU, memory, and log file I/O on both the primary and standby databases are critical to the performance of an asynchronous Data Guard configuration.

To determine which resource is constraining asynchronous transport, use krsb stats which can be enabled by setting event 16421 on both the primary and standby databases:

```
alter session set events '16421 trace name context forever, level 3';
```

This event is very lightweight and won't affect performance of the primary or standby database.

This dynamic event should be set on all primary and standby instances, and it will write statistics into the TT00 or remote file server (RFS) trace file when shipping for a given sequence has completed. Looking in the trace file, you will see the krsb_end stats at the beginning and end of the file. The stats at the end of the file will provide insight into where asynchronous shipping was spending time. For example:

```
krsb_end: Begin stats dump for T-1.S-593
  max number of buffers in use             10
  Operation elapsed time (micro seconds)   474051333
  File transfer time (micro seconds)       474051326
  Network Statistics
   LOG_ARCHIVE_DEST_2 : OCI REQUEST
    Total count  : OCI REQUEST             2748
    Total time   : OCI REQUEST             81374
    Average time : OCI REQUEST             29
   LOG_ARCHIVE_DEST_2 : NETWORK SEND
    Total count  : NETWORK SEND            2748
    Total time   : NETWORK SEND            286554724
    Average time : NETWORK SEND            104277
    Total data buffers queued              9644
    Total data buffers completed           9644
    Total bytes written                    9885272064
    Total bytes completed synchronously    9885272064
    Average network send size (blocks)     7025
    Average network send buffers           3.51
    Average buffer turnaround time         240889
    Throughput (MB/s)                      19.89
   Total network layer time                286636098
   Percentage of time in network           60.47
  Disk Statistics
    Total count  : DISK READ               11531
    Total time   : DISK READ               12335132
    Average time : DISK READ               1069
    Read-ahead blocks                      14151680
    Log buffer blocks                      266915
    Disk stall blocks                      4888576
    Total count  : BUFFER RELEASE          9643
    Total time   : BUFFER RELEASE          7229
    Average time : BUFFER RELEASE          0
   Total disk layer time                   12342361
```

```
  Percentage of time in disk layer        2.60
 Data Guard Processing Statistics
   Total count   : SLEEP                  198
   Total time    : SLEEP                  172351312
   Average time  : SLEEP                  870461
  Total DG layer time                     175072867
  Percentage of time in DG layer          36.93
 Remote Server-Side Network Statistics
  LOG_ARCHIVE_DEST_2 : NETWORK GET
   Total count   : NETWORK GET            8242
   Total bytes   : NETWORK GET            9885272064
   Total time    : NETWORK GET            453233790
   Average time  : NETWORK GET            54990
  Total server-side network layer time    453233790
  Percentage of time in network           95.61
 Remote Server-Side Disk Statistics
  LOG_ARCHIVE_DEST_2 : DISK WRITE
   Total count   : DISK WRITE             9644
   Total time    : DISK WRITE             8731303
   Average time  : DISK WRITE             905
  LOG_ARCHIVE_DEST_2 : DISK NOSTALL REAP
   Total count   : DISK NOSTALL REAP      9644
   Total time    : DISK NOSTALL REAP      579066
   Average time  : DISK NOSTALL REAP      60
  LOG_ARCHIVE_DEST_2 : BUFFER GET
   Total count   : BUFFER GET             9644
   Total time    : BUFFER GET             3607
   Average time  : BUFFER GET             0
  Total server-side disk layer time       9313976
  Percentage of time in disk layer        1.96
 Remote Server-Side Data Guard Processing Statistics
  LOG_ARCHIVE_DEST_2 : PUBLISH RTA BOUNDARY
   Total count   : PUBLISH RTA BOUNDARY   8948
   Total time    : PUBLISH RTA BOUNDARY   3665841
   Average time  : PUBLISH RTA BOUNDARY   409
  LOG_ARCHIVE_DEST_2 : VALIDATE BUFFER
   Total count   : VALIDATE BUFFER        9644
   Total time    : VALIDATE BUFFER        1403088
   Average time  : VALIDATE BUFFER        145
  Total Server-Side DG layer time         11503560
  Percentage of time in DG layer          2.43
krsb_end: End stats dump
```

The above output comes from a test run where a transport lag is just beginning to occur. You can observe a lag due to network congestion increase, and the time waiting on the network layer increases above 50%. If a transport lag is the result of either compression or encryption, the percentage of time spent in the Data Guard layer would become the majority.

To disable krsb stats set event 16421 to level 1:

```
alter session set events '16421 trace name context forever, level 1';
```

Tuning and Troubleshooting Synchronous Redo Transport

Before you proceed, first see Assessing and Optimizing Network Performance.

The following topics describe how to assess synchronous redo transport.

- Understanding How Synchronous Transport Ensures Data Integrity
- Assessing Performance in a Synchronous Redo Transport Environment
- Why the Log File Sync Wait Event is Misleading
- Understanding What Causes Outliers
- Effects of Synchronous Redo Transport Remote Writes
- Example of Synchronous Redo Transport Performance Troubleshooting

Understanding How Synchronous Transport Ensures Data Integrity

The following algorithms ensure data consistency in an Oracle Data Guard synchronous redo transport configuration.

- Log Writer Process (LGWR) redo write on the primary database online redo log and the Data Guard Network Services Server (NSS) redo write to standby redo log are identical.
- The Data Guard Managed Recovery Process (MRP) at the standby database cannot apply redo unless the redo has been written to the primary database online redo log, with the only exception being during a Data Guard failover operation (when the primary is gone).

  In addition to shipping redo synchronously, NSS and LGWR exchange information regarding the safe redo block boundary that standby recovery can apply up to from its standby redo logs (SRLs). This prevents the standby from applying redo it may have received, but which the primary has not yet acknowledged as committed to its own online redo logs.

The possible failure scenarios include:

- If primary database LGWR cannot write to online redo log, then LGWR and the instance crash. Instance or crash recovery will recover to the last committed transaction in the online redo log and roll back any uncommitted transactions. The current log will be completed and archived.
- On the standby, the partial standby redo log completes with the correct value for the size to match the corresponding online redo log. If any redo blocks are missing from the standby redo log, those are shipped over (without reshipping the entire redo log).
- If the primary database crashes resulting in an automatic or manual zero data loss failover, then part of the Data Guard failover operation will do "terminal recovery" and read and recover the current standby redo log.

  Once recovery finishes applying all of the redo in the standby redo logs, the new primary database comes up and archives the newly completed log group. All new and existing standby databases discard any redo in the online redo logs, flashback to a consistent system change number (SCN), and only apply the archives coming from the new primary database. Once again the Data Guard environment is in sync with the (new) primary database.

Assessing Performance in a Synchronous Redo Transport Environment

When assessing performance in an Oracle Data Guard synchronous redo transport environment (SYNC) it is important that you know how the different wait events relate to each other. The impact of enabling synchronous redo transport varies between applications.

To understand why, consider the following description of work the Log Writer Process (LGWR) performs when a commit is issued.

1. Foreground process posts LGWR for commit ("log file sync" starts). If there are concurrent commit requests queued, LGWR will batch all outstanding commit requests together resulting in a continuous strand of redo.
2. LGWR waits for CPU.
3. LGWR starts redo write ("redo write time" starts).
4. For Oracle RAC database, LGWR broadcasts the current write to other instances.
5. After preprocessing, if there is a SYNC standby, LGWR starts the remote write (“SYNC remote write” starts).
6. LGWR issues local write ("log file parallel write").
7. If there is a SYNC standby, LGWR waits for the remote write to complete.
8. After checking the I/O status, LGWR ends "redo write time / SYNC remote write".
9. For Oracle RAC database, LGWR waits for the broadcast ack.
10. LGWR updates the on-disk SCN.
11. LGWR posts the foregrounds.
12. Foregrounds wait for CPU.
13. Foregrounds ends "log file sync".

Use the following approaches to assess performance.

- For batch loads, the most important factor is to monitor the elapsed time, because most of these processes must be completed in a fixed period of time. The database workloads for these operations are very different than the normal OLTP workloads. For example, the size of the writes can be significantly larger, so using log file sync averages does not give you an accurate view or comparison.
- For OLTP workloads, monitor the volume of transactions per second (from Automatic Workload Repository (AWR)) and the redo rate (redo size per second) from the AWR report. This information gives you a clear picture of the application throughput and how it is impacted by enabling synchronous redo transport.

Why the Log File Sync Wait Event is Misleading

Typically, the "log file sync" wait event on the primary database is the first place administrators look when they want to assess the impact of enabling synchronous redo transport (SYNC).

If the average log file sync wait before enabling SYNC was 3ms, and after enabling SYNC was 6ms, then the assumption is that SYNC impacted performance by one hundred percent. Oracle does not recommend using log file sync wait times to measure the impact of SYNC because the averages can be very deceiving, and the actual impact of SYNC on response time and throughput may be much lower than the event indicates.

When a user session commits, the Log Writer Process (LGWR) will go through the process of getting on the CPU, submitting the I/O, waiting for the I/O to complete, and then getting back on the CPU to post foreground processes that the commit has completed. This whole time period is covered by the log file sync wait event. While LGWR is performing its work there are, in most cases, other sessions committing that must wait for LGWR to finish before processing their commits. The size and number of sessions waiting are determined by how many sessions an application has, and how frequently those sessions commit. This batching up of commits is generally referred to as *application concurrency*.

For example, assume that it normally takes 0.5ms to perform log writes (log file parallel write), 1ms to service commits (log file sync), and on average you are servicing 100 sessions for each commit. If there was an anomaly in the storage tier, and the log write I/O for one commit took 20ms to complete, then you could have up to 2,000 sessions waiting on log file sync, while there would only be 1 long wait attributed to log file parallel write. Having a large number of sessions waiting on one long outlier can greatly skew the log file sync averages.

The output from `V$EVENT_HISTOGRAM` for the log file sync wait event for a particular period in time is shown in the following table.

Table 16-1 V$EVENT_HISTOGRAM Output for the Log File Sync Wait Event

Milliseconds	Number of Waits	Percent of Total Waits
1	17610	21.83%
2	43670	54.14%
4	8394	10.41%
8	4072	5.05%
16	4344	5.39%
32	2109	2.61%
64	460	0.57%
128	6	0.01%

The output shows that 92% of the log file sync wait times are less than 8ms, with the vast majority less than 4ms (86%). Waits over 8ms are outliers and only make up 8% of wait times overall, but because of the number of sessions waiting on those outliers (because of batching of commits) the averages get skewed. The skewed averages are misleading when log file sync average waits times are used as a metric for assessing the impact of SYNC.

Understanding What Causes Outliers

Any disruption to the I/O on the primary or standby databases, or spikes in network latency, can cause high log file sync outliers with synchronous redo transport. You can see this effect when the standby system's I/O subsytem is inferior to that of the primary system.

Often administrators host multiple databases such as dev and test on standby systems, which can impair I/O response. It is important to monitor I/O using iostat to determine if the disks reach maximum IOPS, because this affects the performance of SYNC writes.

Frequent log switches are significant cause of outliers. Consider what occurs on the standby when a log switch on the primary occurs, as follows.

1. Remote file server (RFS) process on the standby must finish updates to the standby redo log header.
2. RFS then switches into a new standby redo log with additional header updates.
3. Switching logs forces a full checkpoint on the standby.

   This causes all dirty buffers in the buffer cache to be written to disk, causing a spike in write I/O. In a non-symmetric configuration where the standby storage subsystem does not have the same performance as the primary database, this results in higher I/O latency.
4. The previous standby redo log must be archived, increasing both read and write I/O.

Effects of Synchronous Redo Transport Remote Writes

When you enable synchronous redo transport (SYNC), you introduce a remote write (remote file server (RFS) write to a standby redo log) in addition to the normal local write for commit processing.

This remote write, depending on network latency and remote I/O bandwidth, can make commit processing time increase. Because commit processing takes longer, you observe more sessions waiting on the Log Writer Process (LGWR) to finish its work and begin work on the commit request, that is, application concurrency has increased. You can observe increased application concurrency by analyzing database statistics and wait events.

Consider the example in the following table.

Table 16-2 Affect of Sync Transport Increasing Application Concurrency

SYNC	Redo Rate	Network Latency	TPS from AWR	log file sync average (ms)	log file parallel write average (ms)	RFS random I/O	SYNC remote write average (ms)	Redo write size (KB)	Redo writes
Defer	25MB	0	5,514.94	0.74	0.47	NA	NA	10.58	2,246,356
Yes	25MB	0	5,280.20	2.6	.51	.65	.95	20.50	989,791
Impact	0	-	-4%	+251%	+8.5%	NA	NA	+93.8%	-55.9%

In the above example, enabling SYNC reduced the number of redo writes, but increased the size of each redo write. Because the size of the redo write increased, you can expect the time spent doing the I/O (both local and remote) to increase. The log file sync wait time is higher because there is more work per wait.

However, at the application level, the impact on the transaction rate or the transaction response time might change very little as more sessions are serviced for each commit. This is why it is important to measure the impact of SYNC at the application level, and not depend entirely on database wait events. It is also a perfect example of why log file sync wait event is a misleading indicator of the actual impact SYNC has on the application.

Example of Synchronous Redo Transport Performance Troubleshooting

To look at synchronous redo transport performance, calculate the time spent for local redo writes latency, average redo write size for each write, and overall redo write latency, as shown here.

Use the following wait events to do the calculations.

- local redo write latency = 'log file parallel write'
- remote write latency = ‘SYNC remote write’
- average redo write size per write = ‘redo size’ / ‘redo writes’
- average commit latency seen by foregrounds = 'log file sync'

Statistics from an Automatic Work Repository (AWR) report on an Oracle database are provided in the following table. Synchronous redo transport (SYNC) was enabled to a local standby with a 1ms network latency to compare the performance impact to a baseline with SYNC disabled.

Table 16-3 Assessing Synchronous Redo Transport Performance with Oracle Database

Metric	Baseline (No SYNC)	SYNC	Impact
redo rate (MB/s)	25	25	no change
log file sync	0.68	4.60	+576%
log file parallel write average (ms)	0.57	0.62	+8.8%
TPS	7,814.92	6224.03	-20.3%
RFS random I/O	NA	2.89	NA
SYNC remote write average (ms)	NA	3.45	NA
redo writes	2,312,366	897,751	-61,2%
redo write size (KB)	10.58	20.50	+93.8%

In the above example observe that log file sync waits averages increased dramatically after enabling SYNC. While the local writes remained fairly constant, the biggest factor in increasing log file sync was the addition of the SYNC remote write. Of the SYNC remote write the network latency is zero, so focusing on the remote write into the standby redo log shows an average time of 2.89ms. This is an immediate red flag given that the primary and standby were using the same hardware, and the SYNC remote write average time should be similar to the primary's log file parallel write average time.

In the above example, the standby redo logs have multiple members, and they are placed in a slower performing disk group. After reducing the standby redo logs to a single member, and placing them in a fast disk group, you can see results such as those shown in the following table.

Table 16-4 SYNC Performance After Reducing Standby Redo Logs to a Single Member and Placing on a Fast Disk Group

Metric	Baseline (No SYNC)	SYNC	Impact
redo rate (MB/s)	25	25	no change
log file sync	0.67	1.60	+139%
log file parallel write	0.51	0.63	+23.5%
TPS	7714.36	7458.08	-3.3%
RFS random I/O	NA	.89	NA
SYNC remote write average (ms)	NA	1.45	NA
redo writes	2,364,388	996,532	-57.9%
redo write size (KB)	10.61	20.32	+91.5%

Redo Apply Troubleshooting and Tuning

Most Oracle Data Guard configurations should be able to minimize apply lag by troubleshooting and tuning redo apply. Redo apply performance is directly dependent on the performance of the standby systems.

The guidance presented here assumes that the MAA configuration best practices are followed. As a prerequisites, ensure that the Oracle Data Guard Configuration Best Practices are implemented.

To improve apply performance holistically, leverage the data gathering and troubleshooting methodology described in the topics below.

Understanding Redo Apply and Redo Apply Performance Expectations

Standby database recovery is the process of replaying all DML and DDL operations. The high level process is:

1. Redo is received from the primary database and written into standby redo logs (SRLs). When the database is an Oracle RAC database, each thread (instance) is stored in it's assigned SRLs.
2. The log merger process, sometimes known as the recovery coordinator, merges the threads of redo and places the resulting change vectors into memory buffers.
3. Recovery worker processes identify which data blocks are required and read them into the buffer cache if they are not already present. Then the worker processes apply the change vectors to the blocks in the buffer cache.
4. At checkpoint time, database writer processes write the validated buffer changes to data files, advancing the database's checkpoint time stamp, called the System Commit Number (SCN). Checkpoint can be the most extensive I/O load in the recovery process.

Redo Apply Performance Expectations

Performance, and the resulting apply rate, mainly depend on the type of workload that is being recovered and the system resources allocated to and available for recovery.

Oracle recommends that the primary and standby database systems are symmetric, including equivalent I/O subsystems, memory, and CPU resources. The primary reason for this recommendation is so that the application performs at the same level, no matter which database is the primary database; however, redo apply performance also benefits greatly from symmetric primary and standby databases. Features such as data protection (`DB_BLOCK_CHECKING`, `DB_BLOCK_CHECKSUM`, `DB_LOST_WRITE_PROTECT`) require CPU and I/O resources, as does reporting on the standby database using Oracle Active Data Guard.

For the most part, redo apply performance should keep up with the redo generation rates, resulting in near zero apply lag with system resources are symmetric. During peak workloads, there may be a slight redo apply gap which should naturally reduce to near zero once workloads return to normal levels.

OLTP Workloads

Recovering Online Transaction Processing (OLTP) workloads can be very I/O intensive because an OLTP workload performs small changes to many different blocks. This results in large numbers of small random block reads into the buffer cache during recovery. Subsequently, the database writers run large batches of write I/Os to maintain the buffer cache and to checkpoint the database periodically. Therefore, recovery of OLTP workloads requires the storage subsystem to handle a high number of I/Os Per Second (IOPS) in order to achieve optimal rates. This is another reason for recommending that the primary and standby database systems are symmetric.

Recovery testing of OLTP workloads, generated by swingbench on Oracle Exadata Database Machine quarter rack systems with no resource bottlenecks, achieved approximately 150 MB/sec apply rates. Rates of 200+ MB/s with single instance redo apply have been observed by customers on larger Exadata systems. These rates are more challenging to achieve in non-Exadata systems since the I/O and network throughput are lower.

Batch Workloads

In contrast to OLTP workload recovery, recovering batch workloads is more efficient because batch workloads consist of large sequential reads and writes. A lot more redo changes are occurring while reading and modifying significantly fewer data blocks, resulting in much faster redo apply rates than OLTP workloads. In addition, batch direct load operation recovery optimizations result in greater efficiency and even higher recovery rates.

Using batch load or parallel DML (PDML) workloads with no impeding system resource bottleneck, internal redo apply testing on small Exadata Database Machine quarter rack systems resulted in approximately 200-300 MB/sec apply rates. Customers have observed 600+ MB/sec apply rates with single instance redo apply for their batch workloads for larger Exadata systems. These rates can be achieved by non-Exadata systems, but system resource capacity and scalable network and I/O subsystems are required to handle these demanding workloads.

Mixed Workloads

The difference between OLTP and batch recovery performance profiles and different system shapes explains why applications with variation in their mixtures of OLTP and batch workloads can have different recovery rates at a standby database, even if the primary database redo generation rates are similar. Customers have achieved 100-1100 MB/sec redo apply rates with various mixed workloads for various Exadata systems. These rates can be achieved by non-Exadata systems, but system resource capacity and scalable database compute, network, and I/O subsystems are required to handle these demanding workloads. These extreme redo apply rates are rarely achieved on non-Exadata systems.

Catch Up Redo Apply Performance Expectations

Compared to real-time redo apply, redo apply during a "catch up" period may require even more system resources. If there is a large redo gap, see Addressing a Very Large Redo Apply Gap for recommendations.

Verify Apply Lag

Recovery performance can vary with the workload type and the redo generation rate of the primary database. A lower apply rate does not necessarily indicate a recovery performance issue. However, a persistent or increasing apply lag, without an accompanying transport lag, is the best indication of a recovery performance bottleneck.

To identify and quantify apply lags and transport lags, query the `V$DATAGUARD_STATS` view in the standby database.

```
SQL> select name, value, time_computed, datum_time from v$dataguard_stats
where name='%lag';
```

The `DATUM_TIME` column is the local time on the standby database when the datum used to compute the metric was received. The lag metrics are computed based on data that is periodically received from the primary database. An unchanging value in this column across multiple queries indicates that the standby database is not receiving data from the primary database. The potential data loss in this scenario would be from the last datum time from `V$DATAGUARD_STATS` to the current time on the standby.

To obtain a histogram that shows the history of transport or apply lag values since the standby instance was last started, query the V$STANDBY_EVENT_HISTOGRAM view.

```
 SQL> select * from v$standby_event_histogram where name like '%lag' and
count >0;
```

To evaluate the transport or apply lag over a time period, take a snapshot of V$STANDBY_EVENT_HISTOGRAM in the standby database at the beginning of the time period, and compare that snapshot with one taken at the end of the time period.

```
SQL> col NAME format a10
SQL> select NAME,TIME,UNIT,COUNT,LAST_TIME_UPDATED from
V$STANDBY_EVENT_HISTOGRAM
 where name like '%lag' and count >0 order by LAST_TIME_UPDATED;
```

Example output:

```
NAME       TIME   UNIT       COUNT LAST_TIME_UPDATED

---------- ------ ---------- ----- -------------------
apply lag      23 seconds    3     02/05/2022 16:30:59
apply lag     135 seconds    1     02/05/2022 16:31:02
apply lag     173 seconds    2     02/05/2022 16:32:03
apply lag     295 seconds    2     02/05/2022 16:34:04
```

A transport lag can cause an apply lag. If a high apply lag is observed with a near zero transport lag, continue with this redo apply investigation in Gather Information.

If a high transport lag is observed, first address the transport lag, using the methodology in Redo Transport Troubleshooting and Tuning.

Gather Information

Gather the following information when an unacceptable apply lag is occurring:

- When did the apply lag occur?

  Record the V$DATAGUARD_STATS and V$STANDBY_EVENT_HISTOGRAM data every 15 to 30 minutes to identify when the lag started and how lag changed over time in the last 24 hours.

  ```
  SQL>select name, value, time_computed, datum_time from v$dataguard_stats
  where name='%lag';
  ```

  ```
  SQL>select * from v$standby_event_histogram where name like '%lag' and
  count >0;
  ```

- Does the apply lag occur at certain time period, such as daily at 12 midnight for daily batch operations, monthly during large batch operation, quarterly during quarter end?
- Gather data from the standby Automatic Work Repository (AWR) report V$RECOVERY_PROGRESS, and take multiple standby AWR snapshots at 30 minute intervals before and during the apply lag.

See How to Generate AWRs in Active Data Guard Standby Databases (Doc ID 2409808.1).

For example:

```
SQL> set lines 120 pages 99
SQL> alter session set nls_date_format='YYYY/MM/DD HH24:MI:SS';
SQL> select START_TIME, ITEM, SOFAR, UNITS from gv$recovery_progress;
```

Sample output:

```
START_TIME          ITEM                              SOFAR UNITS
------------------- -------------------------------- ---------- ---------
2022/02/28 23:02:36 Log Files                             8 Files
2022/02/28 23:02:36 Active Apply Rate                 54385 KB/sec
2022/02/28 23:02:36 Average Apply Rate                12753 KB/sec
2022/02/28 23:02:36 Maximum Apply Rate                65977 KB/sec
2022/02/28 23:02:36 Redo Applied                       2092 Megabytes
2022/02/28 23:02:36 Last Applied Redo                     0 SCN+Time
2022/02/28 23:02:36 Active Time                          41 Seconds
2022/02/28 23:02:36 Apply Time per Log                    1 Seconds
2022/02/28 23:02:36 Checkpoint Time per Log               0 Seconds
2022/02/28 23:02:36 Elapsed Time                        168 Seconds
2022/02/28 23:02:36 Standby Apply Lag                     2 Seconds
```

The simplest way to determine application throughput in terms of redo volume is to collect Automatic Workload Repository (AWR) reports on the primary database during normal and peak workloads, and determine the number of bytes per second of redo data the production database is producing. Then compare the speed at which redo is being generated with the Active Apply Rate columns in the V$RECOVERY_PROGRESS view to determine if the standby database is able to maintain the pace.

If the apply lag is above your expectations, then evaluate redo apply performance by querying the V$RECOVERY_PROGRESS view. This view contains the columns described in the following table.

The most useful statistic is the Active Apply rate because the Average Apply Rate includes idle time spent waiting for redo to arrive making it less indicative of apply performance.

Table 16-5 V$RECOVERY_PROGRESS View Columns

Column	Description
Average Apply Rate	Redo Applied / Elapsed Time includes time spent actively applying redo and time spent waiting for redo to arrive
Active Apply Rate	Redo Applied / Active Time is a moving average over the last 3 minutes, and the rate does not include time spent waiting for redo to arrive
Maximum Apply Rate	Redo Applied / Active Time is peak measured throughput or maximum rate achieved over a moving average over last 3 minutes; rate does not include time spent waiting for redo to arrive
Redo Applied	Total amount of data in bytes that has been applied

Table 16-5 (Cont.) V$RECOVERY_PROGRESS View Columns

Column	Description
Last Applied Redo	System change number (SCN) and time stamp of last redo applied. This is the time as stored in the redo stream, so it can be used to compare where the standby database is relative to the primary.
Apply Time per Log	Average time spent actively applying redo in a log file.
Checkpoint Time per Log	Average time spent for a log boundary checkpoint.
Active Time	Total duration applying the redo, but not waiting for redo
Elapsed Time	Total duration applying the redo, including waiting for redo
Standby Apply Lag	Number of seconds that redo apply has not been applied for. Possible standby is behind the primary.
Log Files	Number of log files applied so far.

Active Session History

In cases where standby AWR is not available, or the standby database is not in open read-only mode, the top waits can be gathered using the V$ACTIVE_SESSION_HISTORY view. Standby AWR is strongly recommended due to the additional information and detail provided but these queries are useful in some cases.

To select to top 10 waits over the last 30 minutes (replace 30 with some other number of minutes ago from current time):

```
select * from (
select a.event_id, e.name, sum(a.time_waited) total_time_waited
from v$active_session_history a, v$event_name e
where a.event_id = e.event_id and a.SAMPLE_TIME>=(sysdate-30/(24*60))
group by a.event_id, e.name order by 3 desc)
where rownum < 11;
```

To select the waits between two timestamps (example shows a 3 hour period between 2021/01/01 00:00:00 and 2021/01/01 03:00:00) :

```
select * from (
select a.event_id, e.name, sum(a.time_waited) total_time_waited
from v$active_session_history a, v$event_name e
where a.event_id = e.event_id
and a.SAMPLE_TIME
between to_date('2021/01/01 00:00:00','YYYY/MM/DD HH24:MI:SS') and
to_date('2021/01/01 03:00:00','YYYY/MM/DD HH24:MI:SS')
group by a.event_id, e.name
order by 3 desc)
where rownum < 11
/
```

Compare Redo Generation Rate History on the Primary

There are cases where the primary database's redo generation rate is exceptionally high for a small period of time, such as during large batch jobs, data loads, data pump operations, create table as select or PDML operations or end of month, quarter or year batch updates.

Obtain the redo generation history from the primary database and compare that to when the redo transport or redo apply lag started. Check if the redo generation rate is exceptionally high due to additional workloads, such as adding new pluggable databases (PDBs) or new application services. Additional tuning may be required to accommodate this additional load.

As part of troubleshooting, gather the following information or address the following questions:

- Gather daily history of the primary database's redo generation rate using this query.

```
SQL> select trunc(completion_time) as "DATE", count(*) as "LOG SWITCHES",
round(sum(blocks*block_size)/1024/1024) as "REDO PER DAY (MB)"
from v$archived_log
where dest_id=1
group by trunc(completion_time) order by 1;
```

- Gather the per log redo generation rate, starting 6 hours before the start of any redo or transport lag.

```
SQL> alter session set nls_date_format='YYYY/MM/DD HH24:MI:SS';
SQL> select thread#,sequence#,blocks*block_size/1024/1024 MB,(next_time-
first_time)*86400 sec, blocks*block_size/1024/1024)/((next_time-
first_time)*86400) "MB/s" from v$archived_log
where ((next_time-first_time)*86400<>0)
and first_time between to_date('2015/01/15 08:00:00','YYYY/MM/DD
HH24:MI:SS')
and to_date('2015/01/15 11:00:00','YYYY/MM/DD HH24:MI:SS')
and dest_id=1 order by first_time;
```

- Is this primary redo generation rate exceptionally high compared to prior history?
- If possible, determine the workload that corresponds to the high redo generation rate, and evaluate if it's transient or if it can be tuned.

  For example, for large purge operations, consider truncate or drop partition operations to reduce the redo generation volumes.

Tune Single Instance Redo Apply

Single instance redo apply (SIRA) tuning is an iterative process and a mandatory prerequisite before even evaluating multi-instance redo apply (MIRA). The iterative process consists of

1. Evaluating and addressing system resource bottlenecks
2. Tuning based on top standby database wait events

Evaluate System Resource Bottlenecks

First, evaluate system resources such as CPU utilization and I/O subsystem. Use utilities such as `top` and `iostat` or statistics from OSwatcher or ExaWatcher to determine if there is

contention for those resources. Addressing any resource bottlenecks to free up resources required for redo apply can improve apply performance.

Redo apply can be impacted if:

- The managed recovery node is completely CPU bound
- The standby database's I/O system is saturated
- The standby database SGA, specifically the buffer cache, is not at least the same size (or larger) than that on the primary database

For optimal recovery performance the standby database system requires:

- Sufficient CPU utilization for Recovery Coordinator (PR00) and recovery workers (PR*nn*)
- Sufficient I/O bandwidth to maintain low I/O latency during peak rates
- A network interface that can receive the peak redo rate volumes, in addition to any other network activity across the same interface
- Sufficient memory to accommodate a symmetric SGA and buffer cache; the size of the log buffer and buffer cache generally have the biggest impact on redo apply performance

What to gather and how?

- Gather standby Automatic Work Repository (AWR) reports with intervals of 30 minutes or less.

  See Managing Automatic Workload Repository in Active Data Guard Standby Databases in *Oracle Database Performance Tuning Guide*
- Gather Active Session History (ASH) data for more real time granular waits.

  See Generating Active Session History Reports in *Oracle Database Performance Tuning Guide*
- Gather Oracle Linux OSwatcher or Oracle Exadata ExaWatcher data to analyze system resources.

  For Exadata systems, see Using ExaWatcher Charts to Monitor Exadata Database Machine Performance in *Oracle Exadata Database Machine Maintenance Guide*
- Gather `top` process information to check if the recovery coordinator (PR00) is CPU bound by using `top` or `ps` commands.

Some common indicators and causes of resource bottlenecks include:

- Low CPU idle time may indicate the system is CPU bound
- Long disk or flash service times or high IOPS may indicate I/O contention or saturation
- Undersized systems and shared systems with many active databases may cause contention for these resources
- Reporting workloads in an Active Data Guard standby can also cause contention

Tune Redo Apply by Evaluating Database Wait Events

Once you have verified that there are no system resource bottlenecks, it is time to assess standby database wait events by looking at the standby Automatic Work Repository (AWR) reports.

Before assessing database wait events, it is important to understand where the waits occur during the process flow involved in recovery.

1. Redo is received on the standby by the Remote File Server (RFS) process.

The RFS process writes newly received redo for each thread into the current standby redo log for that thread. The RFS write operation is tracked by the rfs random I/O wait event.

2. Once redo has been written, the recovery coordinator process (pr00) reads the redo from the standby redo logs (or archived logs) for each thread.

   This read I/O is tracked by the log file sequential read wait event.

3. The recovery coordinator then merges redo from all threads together and places the redo into memory buffers for the recovery workers.

   The wait events for writing and reading into recovery memory buffers is tracked by the parallel recovery read buffer free and parallel recovery change buffer free wait events.

4. The recovery processes retrieve redo or change vectors from the memory buffers and begin the process of applying the changes to data blocks.

   First the recovery workers determine which data blocks need to be recovered and reads those into the buffer cache if it's not already present.

   This read I/O by the recovery workers is tracked by the recovery read wait event.

5. When a log is switched on the primary for any thread, the standby coordinates a switch of the standby redo log for that thread at the same time.

   In earlier versions a log switch on a standby forces a full checkpoint, which results in flushing all dirty buffers from the buffer cache out to the data files on the standby. Starting with Oracle Database 18c, checkpoints also occur at regular time intervals, thus amortizing checkpoint I/O across all phases.

   During checkpoint, multiple database writer processes (DBWR) write the data file blocks down to the data files, with its write time tracked by the db file parallel write wait event. The total time for the checkpoint to complete is covered by the checkpoint complete wait event.

During the apply phase it is normal to observe that the recovery coordinator process (pr00) has high utilization on a single CPU, while during the checkpoint phase there is an increase in DB writer processes (dbw*n*) CPU utilization indicating increased write I/O to the data files.

The following table provides a description as well as tuning advice for wait events involved in the recovery process.

Table 16-6 Recovery Process Wait Events

Column	Description	Tuning Recommendations
Logfile sequential read	The parallel recovery coordinator is waiting on I/O from the online redo log, standby redo log, or the archived redo log.	Tune or increase the I/O bandwidth for the ASM disk group or storage subsystem where the archive logs, standby redo logs, or online redo logs reside.
Parallel recovery read buffer free	This event indicates that all read buffers are being used by workers, and usually indicates that the recovery workers lag behind the coordinator.	Increase `_log_read_buffers` to max 256
Parallel recovery change buffer free	The parallel recovery coordinator is waiting for a buffer to be released by a recovery worker. Again, this is a sign the recovery workers are behind the coordinator.	Tune or increase the I/O bandwidth for the ASM disk group or storage subsystem where data files reside.

Table 16-6 (Cont.) Recovery Process Wait Events

Column	Description	Tuning Recommendations
Data file init write	The parallel recovery coordinator is waiting for a file resize to finish, as would occur with file auto extend.	This is a non-tunable event, but evaluate your primary database workload to discover why there are so many data file resize operations. Optionally, use a larger NEXT size when AUTOEXTEND is enabled
Parallel recovery control message reply	The parallel recovery coordinator is waiting for all recovery workers to respond to a synchronous control message.	N/A. This is an idle event.
Parallel recovery slave next change	The parallel recovery worker is waiting for a change to be shipped from the coordinator. This is in essence an idle event for the recovery worker. To determine the amount of CPU a recovery worker is using, divide the time spent in this event by the number of workers started, and subtract that value from the total elapsed time.	N/A. This is an idle event.
DB File Sequential Read	A parallel recovery worker (or serial recovery process) is waiting for a batch of synchronous data block reads to complete.	Tune or increase the I/O bandwidth for the ASM disk group or storage subsystem where data files reside.
Checkpoint completed	Recovery is waiting for checkpoint to complete, and redo apply is not applying any changes currently.	Tune or increase the I/O bandwidth for the ASM disk group or storage subsystem where data files reside. Also, increase the number of `db_writer_processes` until the checkpoint completed wait event is lower than the db file parallel write wait event. Also consider increasing the online log file size on the primary and standby to decrease the number of full checkpoints at log switch boundaries.
Recovery read	A parallel recovery worker is waiting for a batched data block I/O.	Tune or increase the I/O bandwidth for the ASM disk group where data files reside.

Table 16-6 (Cont.) Recovery Process Wait Events

Column	Description	Tuning Recommendations
Recovery apply pending and/or recovery receive buffer free (MIRA)	Recovery apply pending = the time the logmerger process waited (in centiseconds) for apply workers to apply all pending changes up to a certain SCN. Recovery receive buffer free = the time (in centiseconds) spent by the receiver process on the instance waiting for apply workers to apply changes from received buffers so that they can be freed for the next change.	Increase `_mira_num_local_buffers` and `_mira_num_receive_buffers` Note that these parameters use space from the shared pool equal to the sum of their values (in MB) multiplied by the number of apply instances.

See How to Generate AWRs in Active Data Guard Standby Databases (Doc ID 2409808.1) for more information about generating AWRs on the standby database.

Enable Multi-Instance Redo Apply if Required

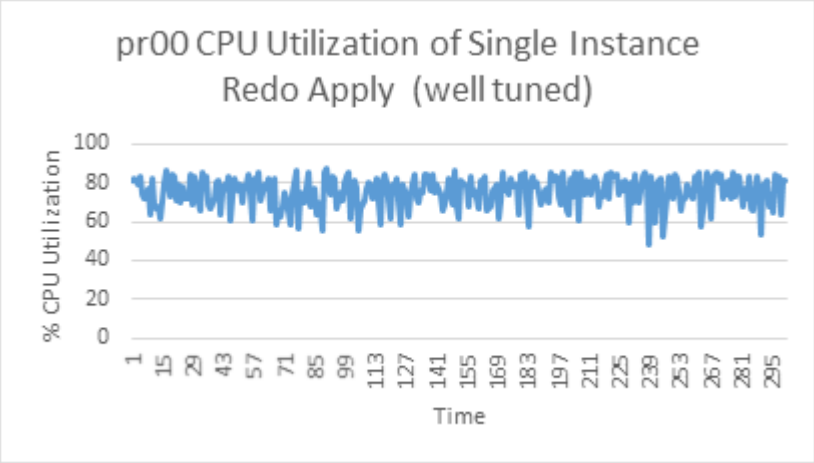
Multi-instance redo apply (MIRA) has the potential to improve redo apply by running multiple recovery coordinators and redo apply (worker) processes across Oracle RAC database instances of the standby database. MIRA is optimized for later Oracle Database releases, and the redo apply benefits vary based on workloads.

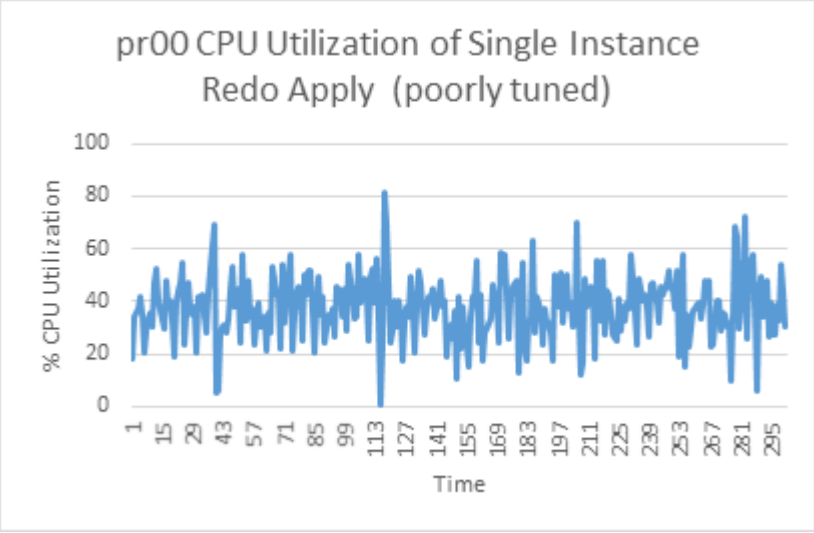
Prerequisites for Considering MIRA

- Single-instance redo apply (SIRA) has been completely tuned and is not I/O bound.
- Recovery coordinator (PR00) is CPU bound.

  Examine the CPU utilization of the recovery coordinator/log merger process *ora_pr00_<SID>* over a period of an hour. If the coordinator process has a CPU utilization % of over 70% for a majority of that time, this may be the bottleneck, and MIRA may improve recovery performance.

  Shown here are two examples of output from the `top` command showing the CPU utilization of the pr00.





If the recovery coordinator CPU utilization is largely below 70% with only a few short spikes, it is not CPU bound, and there is likely a resource issue or some additional tuning that will improve performance. If the recovery coordinator is not CPU bound, return to tuning SIRA.

- Most MIRA optimizations are implemented in Oracle Database 19c and are not available in earlier database releases. In fact, Oracle recommends the database release be no earlier than Oracle Database 19.13 because it includes some important fixes, including 29924147, 31290017, 31047740, 31326320, 30559129, 31538891, 29785544, 29715220, 29845691, 30421009, 30412188, 30361070, 32486528, 33821145 and 28389153.
- All Oracle Exadata Database Machine systems based either on InfiniBand network fabric or on RDMA over Converged Ethernet (RoCE) network fabric require an additional step on the primary database, as shown in this table.

Table 16-7 Oracle Exadata Database Machine prerequisites to enable MIRA

Exadata System	Database Release	Steps
Exadata Storage cells with persistent memory (PMEM)	19.13 and higher	No additional steps
Without PMEM	19.13 and higher	Set dynamic parameter on all instances `_cache_fusion_pipelined_updates_enable=FALSE`

Table 16-7 (Cont.) Oracle Exadata Database Machine prerequisites to enable MIRA

Exadata System	Database Release	Steps
Any Exadata System	19.12 and lower	1. Apply Patch 31962730 2. Set dynamic parameter on all instances `_cache_fusion_pipelined_updates_enable=FALSE`

> **Note:**
>
> Only redo generated with the dynamic parameter `_cache_fusion_pipelined_updates_enable` or static parameter `_cache_fusion_pipelined_updates` set to FALSE can be recovered with MIRA.

Enable Multi-instance Redo Apply and Tune

1. Enable multi-instance redo apply (MIRA) by indicating the number of apply instances.

   Leave all previous single-instance redo apply (SIRA) tuning changes in place. The MAA recommendation for MIRA is to use all standby database instances for apply.

2. Enable MIRA using one of these methods.

   - Set an Oracle Data Guard Broker property

     ```
     'ApplyInstances'=<#|ALL>
     ```

   - Or run

     ```
     SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT FROM
     SESSION INSTANCES ALL;
     ```

3. Check for system resource contention after tuning MIRA.

   Follow the same practices described in Evaluate System Resource Bottlenecks.

4. Tune MIRA based on wait events described here.

   Follow the methodology in Tune Redo Apply by Evaluating Database Wait Events.

   If recovery apply pending or recovery receive buffer free are among the top wait events:

   - Increase `_mira_num_receive_buffers` and `_mira_num_local_buffers` incrementally by 100 to reduce this wait event.

     These parameters provide additional buffer space to pass blocks between instances. Evaluate whether there is sufficient memory in the SGA to accommodate the additional buffer space.

     The additional memory requirements for each participating MIRA Oracle RAC instance = (_mira_num_receive_buffers + _mira_num_local_buffers) * (# of RAC instances * 2) MB

     For example, if _mira_num_receive_buffers=500 and _mira_num_local_buffers=500, then (500+500) *(4-node RAC *2) = 8000MB from the SGA

   - Set `_mira_rcv_max_buffers=10000`

Addressing a Very Large Redo Apply Gap

If the apply lag is larger than 24 hours, consider using a standby roll forward method to skip over the gap rather than apply all of the redo. See How to Roll Forward a Standby Database Using Recover Database From Service (12.2 and higher) (Doc ID 2850185.1)

This approach pulls changed Oracle data blocks directly from the primary database, and can potentially mitigate a large redo gap in half the time required to apply all of the redo.

The disadvantages of this approach are:

- Logical corruption and lost write detection checks and balances that are inherent to redo apply and standby databases are skipped
- Manual intervention is required to issue these commands and restart redo apply once it's completed.

Data blocks are still verified for physical corruptions.

Improving Redo Apply Rates by Sacrificing Data Protection

There are extremely rare conditions where redo apply cannot be tuned to achieve even higher redo apply rates to stay current with the primary. In these cases it may be necessary to turn off recommended data protection settings to help improve redo apply performance.

The following table describes some potential interim changes and their potential gains and trade offs.

Change	Potential Gain	Potential Trade-offs
Stop redo apply and use recover from service See How to Roll Forward a Standby Database Using Recover Database From Service (12.2 and higher) (Doc ID 2850185.1)	Optimized approach to recover from a large redo transport or redo apply gap, such as when the gap exceeds 24 hours	No logical block or lost writes data protection checks No redo block checksum verification
Mount standby instead of Active Data Guard	Potential 5-10% redo apply performance gain, but mostly for batch workloads	No real-time auto block repair of physical corruptions No real-time query on the standby Neither of the above trade-offs may be as relevant when the standby is lagging beyond application threshold
Disable or reduce `DB_BLOCK_CHECKING` on the standby	Reduces CPU utilization during redo apply If CPU resources are limited, this change can improve redo apply by 10-40%	Potential "rare" logical block corruptions may not be detected and can be propagated to the standby
Disable Flashback Database	Eliminates flashback IOPS requirement on RECO If storage IOPS is the constraining resource, then this change can help redo apply performance	Lose the ability to quickly rewind the standby
Disable `DB_LOST_WRITE_PROTECT` on the primary and standby	Eliminates additional read IOPS on the standby due to block read redo generated on the primary to detect lost writes This change is an option if IOPS capacity is saturated	Lost writes are not detected early on either primary or standby databases

Role Transition, Assessment, and Tuning

With thorough planning, configuration, and tuning, Oracle Data Guard role transitions can effectively minimize downtime and ensure that the database environment is restored with minimal impact on the business.

Using a physical standby database, Oracle MAA testing has determined that switchover and failover times with Oracle Data Guard have been reduced to seconds. This section describes best practices for both switchover and failover. While following best practices, switchover times of approximately 30 seconds for Oracle RAC and less 10 seconds for a single instance database have been observed. Detection time is separate.

Prerequisite Data Guard Health Check Before Role Transition

Complete the following prerequisites before performing a switchover operation.

Every Quarter

Perform the following steps every quarter.

1. Ensure that your Oracle Data Guard configuration is MAA compliant.
   a. Refer to Oracle Database Configuration Best Practices and Oracle Data Guard Configuration Best Practices to ensure that all recommended Data Guard configuration practices are in place.
   b. Refer to Overview of Oracle Multitenant Best Practices for PDB service recommendations.
2. Run a simple application test, which includes:
   a. Convert existing the standby database to a snapshot standby.
   b. Validate the application connection to the read-write test database as if this was a disaster recovery test. See Configuring Continuous Availability for Applications for configuration guidance.
3. Test your end-to-end application failover after a Data Guard role transition.
   a. Issue a Data Guard switchover.
   b. Orchestrate the entire application failover.
   c. Switch back is optional.

One Month Before Switchover

One month before performing a switchover operation, consult the MOS note “Oracle Database 19c Important Recommended One-off Patches (Doc ID 555.1)” to identify any critical issues that might affect your release.

Also consider suspending or shutting down long running reporting or jobs including monitoring, auditing, and database backups that create persistent connections during the target planned maintenance window that contains the Data Guard switchover operation.

Common configuration issues that impact application service availability while performing a Data Guard role transition with Oracle Multitenant database are:

- PDB saved state or triggers are used and fail during Data Guard role transition

- PDB default service is leveraged instead of using Oracle clusterware-managed distinct services for each PDB for your application service
- Wallet/security settings are not the same on the standby

To ensure application service and application failover readiness:

1. Never use PDB default services, nor `SAVED STATE` (except during relocate operations), nor database triggers to manage role-based services.
2. Use clusterware-managed distinct services on each PDB for your application service, and leverage that application service to connect to the database.
3. When defining a clusterware-managed application service, define which PDB and services will be started, and in which Oracle RAC instance and database role.
4. For Data Guard, always use role-based services by assigning a role to each clusterware-managed service.

Validate Database Switchover and Failover Readiness

You can use the `VALIDATE` command to perform a comprehensive set of database checks before performing a role change. The command checks the following items:

- Whether there is missing redo data on a standby database
- Whether flashback is enabled
- The number of temporary tablespace files configured
- Whether an online data file move is in progress
- Whether online redo logs are cleared for a physical standby database
- Whether standby redo logs are cleared for a primary database
- The online log file configuration
- The standby log file configuration
- Apply-related property settings
- Transport-related property settings
- Whether there are any errors in the Automatic Diagnostic Repository (for example, control file corruptions, system data file problems, user data file problems)

The three main `VALIDATE` commands that should be issued prior to switchover are:

1. `VALIDATE DATABASE VERBOSE` *`standby`* - The `VALIDATE DATABASE` command shows a brief summary of the database, and reports any errors or warnings that were detected. `VALIDATE DATABASE VERBOSE` shows everything in the brief summary plus all items that were validated.
2. `VALIDATE DATABASE` *`standby`* `SPFILE` - The `VALIDATE DATABASE SPFILE` command reports any parameter differences between primary and the specified standby databases.
3. `VALIDATE NETWORK CONFIGURATION FOR ALL` - The `VALIDATE NETWORK CONFIGURATION` command performs network connectivity checks between members of a configuration.

To summarize how to evaluate Role Transition readiness, review the following:

- `PRIMARY DATABASE` Section:
  - `DGMGRL> VALIDATE DATABASE VERBOSE '`*`Primary_DBName`*`';`
  - Check if there are PDB saved states in the primary database.

    - `SELECT * FROM dba_pdb_saved_states;`
  - Evaluate health with `exachk` or `orachk`.
- For each `STANDBY DATABASE` *`STANDBY_DB_UNIQUE_NAME`* Section:
  - `DGMGRL> VALIDATE DATABASE VERBOSE '`*`Standby_DBName`*`';`
  - `DGMGRL> VALIDATE DATABASE '`*`Standby_DBName`*`' SPFILE;`
  - Evaluate health with `exachk` or `orachk`.
  - Evaluate if the standby cluster and database are symmetric with the primary cluster and database. This ensures identical or similar performance after role transition.
  - Evaluate whether the cluster shape and system resources are the same, spfile memory settings are the same, and number of databases sharing the cluster resources are the same. If not, highlight the differences and evaluate if system resources are available by reviewing `exawatcher` or `oswatcher` graphs.
- Network Section:
  - `DGMGRL> VALIDATE NETWORK CONFIGURATION FOR ALL;`
- Redo Rate History Section:
  - ```
    SQL> SELECT thread#,sequence#,blocks*block_size/1024/1024 MB,(next_time-
    first_time)*86400 sec,
     blocks*block_size/1024/1024)/((next_time-first_time)*86400) "MB/s"
     FROM v$archived_log
     WHERE ((next_time-first_time)*86400<>0) and first_time
     between to_date('2015/01/15 08:00:00','YYYY/MM/DD HH24:MI:SS')
     and to_date('2015/01/15 11:00:00','YYYY/MM/DD HH24:MI:SS') and
    dest_id=1
     order by first_time;
    ```

Example:

The Oracle Data Guard broker `VALIDATE DATABASE` command gathers information related to switchover and failover readiness.

The validation verifies that the standby and primary database are reachable and the apply lag is less than `ApplyLagThreshold` for the target database. If these data points are favorable, the command output displays "Ready for Failover: Yes" as shown below. In addition, if redo transport is running, the command output displays "Ready for Switchover: Yes".

```
DGMGRL> validate database [verbose] database_name

Database Role: Physical standby database
 Primary Database: standby_db_unique_name

Ready for Switchover: Yes
 Ready for Failover: Yes (Primary Running)
```

`VALIDATE DATABASE` checks additional information that can impact switchover time and database performance, such as whether the online redo logs have been cleared, number of
```

temporary tablespaces, parameter mismatches between primary and standby, and the status of flashback databases.

In most failover cases the primary database has crashed or become unavailable. The Ready for Failover output indicates if the primary database is running when `VALIDATE DATABASE` was issued. This state does not prevent a failover, but it is recommended that you stop the primary database before issuing a failover to avoid a *split-brain* scenario where the configuration has two primary databases. The broker only guarantees split-brain avoidance on failover when Fast-Start Failover is used.

You should also run `VALIDATE DATABASE VERBOSE` *`standby`*, `VALIDATE DATABASE` *`standby`* `SPFILE`, and `VALIDATE NETWORK CONFIGURATION FOR ALL` periodically as a configuration monitoring tool.

Days Before Switchover

Perform the following steps days before performing a Data Guard switchover.

1. **Set the Data Guard broker trace level.**

   The Data Guard broker `TraceLevel` configuration property is used to control the amount of tracing performed by the broker for every member in the configuration. Setting the property to `USER` limits the tracing to completed operations and to any warning or error messages resulting from an operation or health check. Setting the property to `SUPPORT` increases the amount of tracing to include lower-level information needed to troubleshoot any issues.

   ```
   DGMGRL> SET TRACE_LEVEL SUPPORT;
   ```

2. **Enable role transition metrics.**

   The Time Management Interface (TMI) event is a low overhead event which adds a line to the alert log whenever certain calls are executed in Oracle.

   These entries in the alert log, or tags, delineate the beginning and end of a call. The tables in the topics below depict the delineation of key switchover and failover operations. This method is the most accurate for determining where time is being spent.

   Set the database level event 16453 trace name context forever, level 15 on all databases. There are two methods of enabling this trace, either using the `EVENT` database parameter or setting the `EVENTS` at the system level. The difference is that the `EVENT` parameter is not dynamic but is persistent across restarts. `SET EVENTS` is dynamic but NOT persistent across database restarts. See the following examples.

   ```
   ALTER SYSTEM SET EVENT='16453 trace name contextforever, level 15'
   scope=spfile sid='*';


   ALTER SYSTEM SET EVENTS '16453 trace name context forever, level 15';
   ```

Data Guard Role Transition

Always use Oracle Data Guard broker or any Oracle UI or utility that ultimately calls the Data Guard broker command.

Suspend or shut down any long running reports or batch jobs including monitoring, auditing, and database backups that have persistent connections.

Use the Oracle Data Guard broker `SWITCHOVER` command to initiate switchover, and the `FAILOVER` command to initiate failover.

As part of a switchover or failover operation the broker does the following.

- Configures redo transport from the new primary database
- Starts redo apply on the new standby database
- Ensures that other standby databases in the broker configuration are viable and receiving redo from the new primary
- Integrates Oracle Clusterware and Global Data Services to ensure that the role-based services are started

Before issuing the Data Guard switchover, suspend or shut down long running reporting or jobs including monitoring, auditing, and database backups that create persistent connections.

To configure broker to initiate switchover, log in as SYS or SYSDBA and issue:

```
DGMGRL> SWITCHOVER TO database_name;
```

To configure broker to initiate failover, run:

```
DGMGRL> FAILOVER TO database_name [IMMEDIATE];
```

By default `FAILOVER` applies all redo that was received before failing over. The `IMMEDIATE` clause skips the pending redo and fails over immediately.

The `SWITCHOVER` and `FAILOVER` commands are idempotent and can be re-issued in the unlikely event of a failed transition.

Monitor Data Guard Role Transitions

Refer to the Data Guard Broker messages while the Data Guard role transition is happening. To extract detailed role transition status, refer to the primary and standby alert logs and broker logs for Data Guard switchover and failover messages and tags.

Key Switchover Operations and Alert Log Tags

Switchover is broken down into four main steps as follows.

1. `Convert to Standby` - terminate any existing production sessions, convert the control file into a standby control file, and send a message to the standby to continue the switchover.

   The `Convert to Standby` - these steps are found in the alert log of the original primary. All remaining steps are found in the original standby alert log.

2. `Cancel Recovery` - apply remaining redo and stop recovery.
3. `Convert to Primary` - a two-step close (to the mounted state) of instances (one instance, then all others), clear online redo logs, convert control file to primary control file, and data Guard Broker bookkeeping.
4. `Open New Primary` - parallel open of all instances.

Table 16-8 Alert Log Tags Defining the Steps with Time Management Interface Event Enabled

Step	Stage	Time Management Interface Event Enabled
Convert To Standby(primary alert log)	BEGIN	TMI: dbsdrv switchover to target BEGIN <DATE> <TIMESTAMP>
Convert To Standby(primary alert log)	END	TMI: kcv_switchover_to_target send 'switchover to primary' msg BEGIN <DATE> <TIMESTAMP>
Cancel Recovery(standby alert log)	BEGIN	TMI: kcv_commit_to_so_to_primary wait for MRP to die BEGIN <DATE> <TIMESTAMP>
Cancel Recovery(standby alert log)	END	TMI: kcv_commit_to_so_to_primary wait for MRP to die END <DATE> <TIMESTAMP>
Convert to Primary (standby alert log)	BEGIN	TMI: kcv_commit_to_so_to_primary BEGIN CTSO to primary <DATE> <TIMESTAMP>
Convert to Primary (standby alert log)	END	TMI: adbdrv BEGIN 10 <DATE> <TIMESTAMP>
Open Primary(standby alert log)	BEGIN	TMI: adbdrv BEGIN 10 <DATE> <TIMESTAMP>
Open Primary(standby alert log)	END	TMI: adbdrv END 10 <DATE> <TIMESTAMP>

Key Failover Operations and Alert Log Tags

All failover steps are documented in the alert log of the target standby where the failover was performed.

1. Cancel Recovery - Stop recovery and close all instances (to mounted) in parallel.
2. Terminal Recovery - Archive standby redo logs and recover any unapplied redo.
3. Convert to Primary - Clear online redo logs and convert control file to standby control file.
4. Open Primary - Open all instances in parallel.

Table 16-9 Failover Alert Log Tags Defining the Steps with Time Management Interface Event Enabled

Step	Stage	Time Management Interface Event Enabled
Cancel Recovery	BEGIN	TMI: adbdrv termRecovery BEGIN <DATE> <TIMESTAMP>
Cancel Recovery	END	TMI: adbdrv termRecovery END <DATE> <TIMESTAMP>
Terminal Recovery	BEGIN	TMI: krdsmr full BEGIN Starting media recovery <DATE> <TIMESTAMP>

Table 16-9 (Cont.) Failover Alert Log Tags Defining the Steps with Time Management Interface Event Enabled

Step	Stage	Time Management Interface Event Enabled
Terminal Recovery	END	TMI: krdemr full END end media recovery <DATE> <TIMESTAMP>
Convert to Primary	BEGIN	TMI: kcv_commit_to_so_to_primary BEGIN CTSO to primary <DATE> <TIMESTAMP>
Convert to Primary	END	TMI: adbdrv BEGIN 10 <DATE> <TIMESTAMP>
Open Primary	BEGIN	TMI: adbdrv BEGIN 10 <DATE> <TIMESTAMP>
Open Primary	END	TMI: adbdrv END 10 <DATE> <TIMESTAMP>

Post Role Transition Validation

Use the `SHOW CONFIGURATION VERBOSE` command to verify that the switchover or failover and standby reinstate was successful.

```
DGMGRL> SHOW CONFIGURATION VERBOSE;
Configuration - DRSolution
Protection Mode: MaxAvailability
Members:
         South_Sales  - Primary database
         North_Sales - Physical standby database
         Fast-Start Failover: DISABLED
         Configuration Status:
          SUCCESS
```

Troubleshooting Problems During a Switchover Operation

The most important goal after a failed Data Guard switchover or failover operation is to resume database and application availability as soon as possible.

Sources of Diagnostic Information

The Oracle Data Guard broker provides information about its activities in several forms.

- Database status information - You can use the `SHOW DATABASE VERBOSE` *`db_unique_name`* command to get a brief description of the database (name, role, and so on), database status, and information about any health check problems.

  ```
  DGMGRL> SHOW DATABASE VERBOSE db_unique_name
  ```

- Oracle alert log files - The broker records key information in the alert log file for each instance of each database in a broker configuration. You can check the alert log files for such information when troubleshooting Oracle Data Guard.

- Oracle Data Guard "broker log files" - For each instance of each database in a broker configuration, the broker DMON process records important behavior and status information in a broker log file, useful in diagnosing Oracle Data Guard failures. The `TraceLevel` configuration property is used to specify the level of diagnostic information reported in the broker log files. The broker log file is created in the same directory as the alert log and is named `drc<$ORACLE_SID>.log`.

Retry Switchover After Correcting the Initial Problem

If the reported problems can be corrected quickly, you can retry the switchover operation.

If the reported problems cannot be corrected or the switchover operation fails even after correcting the reported problems, then you can choose another database for the switchover or restore the configuration to its pre-switchover state and then retry the switchover or refer to Rolling Back After Unsuccessful Switchover to Maximize Uptime.

```
DGMGRL> SWITCHOVER TO database_name;
```

Rolling Back After Unsuccessful Switchover to Maximize Uptime

For physical standby databases in situations where an error occurred, and it is not possible to continue with the switchover in a timely fashion, revert the new physical standby database back to the primary role to minimize database downtime.

Take the following steps.

1. Shut down and mount the new standby database (old primary).
2. Start Redo Apply on the new standby database.
3. Verify that the new standby database is ready to be switched back to the primary role.

   Query the `SWITCHOVER_STATUS` column of the `V$DATABASE` view on the new standby database. A value of `TO PRIMARY` or `SESSIONS ACTIVE` indicates that the new standby database is ready to be switched to the primary role. Continue to query this column until the value returned is either `TO PRIMARY` or `SESSIONS ACTIVE`.
4. Issue the following statement to convert the new standby database back to the primary role:

   ```
   SQL> ALTER DATABASE SWITCHOVER TO target_db_name;
   ```

   If step 4 fails, see Roll Back After Unsuccessful Switchover and Start Over in

Data Guard Performance Observations

Data Guard Role Transition Duration

Oracle Data Guard and Oracle MAA Gold reference architectures provide disaster recovery and high availability solutions when the primary database, cluster, or site fails or is inaccessible.

Each Data Guard environment is different and the time to perform role transitions can vary significantly. Variables including, but not limited to, SGA size, number of Oracle RAC instances,

number of PDBs, data files, and connections to the database at the time of role transition impact the length of a given role transition.

Generally, Data Guard switchover (planned maintenance) is slightly longer than Data Guard failover (unplanned outages).

The following information is meant to educate you about ways to optimize role transitions.

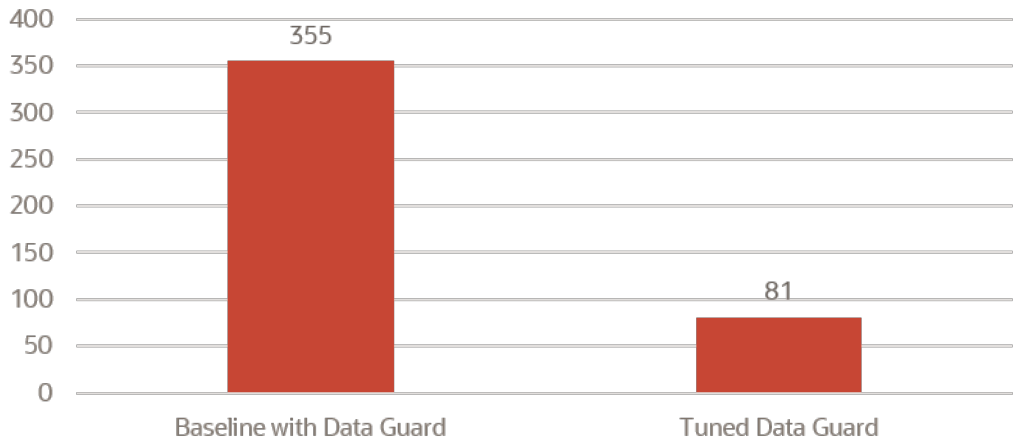
Data Guard Switchover Duration

When attempting to minimize application downtime for planned maintenance:

- Before planned maintenance windows, avoid or defer batch jobs or long running reports. Peak processing windows should also be avoided.
- Because Data Guard switchover is graceful, which entails a shutdown of the source primary database, any application drain timeout is respected. See Enabling Continuous Service for Applications for Oracle Clusterware service drain attributes and settings.
- Data Guard switchover operations on a single instance (non-RAC) can be less than 30 seconds.
- Data Guard switchover operations on Real Application Cluster vary, but can be from 30 seconds to 7 minutes. The duration may increase with more PDBs (for example, > 25 PDBs), more application services (for example, 200 services), and if the database has a large number of data files (for example, 1000s of data files).

The following graph and table show one example of how much switchover operation duration can decrease when MAA tuning recommendations are implemented. Results will vary.

Figure 16-1 Planned maintenance: DR switch duration in seconds



Planned DR Switch (Switchover)	Initial Configuration	Tuned MAA Configuration
Convert Primary to Standby	26 secs	21 sec
Convert Standby to Primary (C2P)	47 secs	7 secs
Open new Primary (OnP)	152 secs	14 secs
Open PDB and Start Service (OPDB)	130 secs	39 secs

Planned DR Switch (Switchover)	Initial Configuration	Tuned MAA Configuration
Total App Downtime	**355 secs or 5 mins 55 secs**	**81 secs (78% drop)**

The "Tuned" timings were achieved by implementing the following MAA recommended practices:

- Use Bigfile Tablespace
- Oracle Data Guard Configuration Best Practices
- Role Transition, Assessment, and Tuning

Data Guard Failover Duration

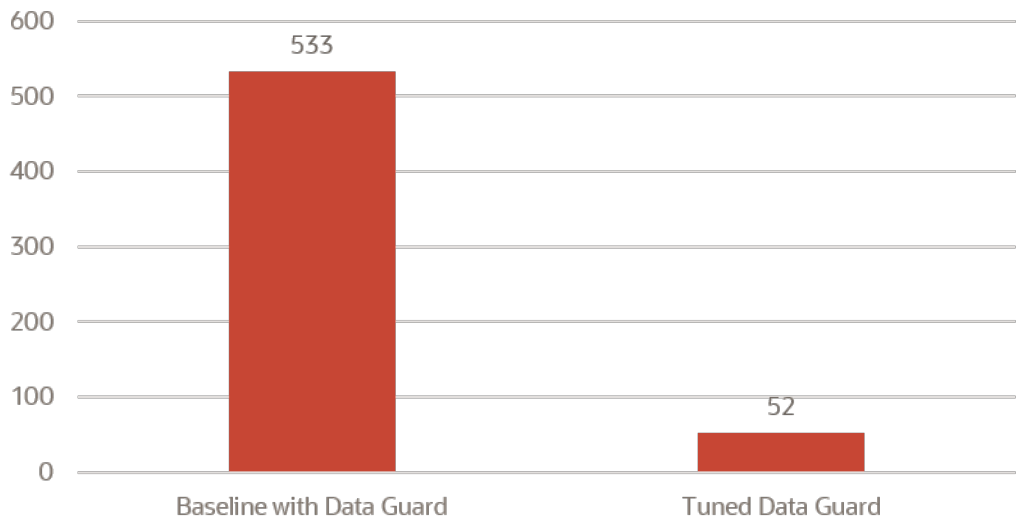
When attempting to minimize application downtime for DR scenarios:

- To limit Recovery Time Objective (RTO or database down time) and Recovery Point Objective (RPO or data loss), automatic detection and fail over is required. See Fast-Start Failover in *Oracle Data Guard Broker*.
- The database administrator can determine the appropriate "detection time" before initiating an automatic failover by setting `FastStartFailoverThreshold`. See Enabling Fast-Start Failover Task 4: Set the FastStartFailoverThreshold Configuration Property in *Oracle Data Guard Broker*.

  The MAA recommended setting is between 5 seconds and 60 seconds for a reliable network. Oracle RAC restart may also recover from a transient error on the primary. Setting this threshold higher gives the restart a chance to complete and avoid failover, which can be intrusive in some environments. The trade off is that application downtime increases in the event an actual failover is required.
- Data Guard failover operations on a single instance (non-RAC) can be less than 20 seconds.
- Data Guard failover operations on a Real Application Cluster vary but can be from 20 seconds to 7 minutes. The duration may increase with more PDBs (for example, > 25 PDBs), more application services (for example, 200 services) and if the database has a large number of data files (for example, 1000s of data files).

The following graph and table show one example how much failover operation duration can decrease when MAA tuning recommendations are implemented. Results will vary.

Figure 16-2 Unplanned DR failover duration in seconds



Unplanned Outage/DR (Failover)	Initial Configuration	Tuned MAA Configuration
Close to Mount (C2M)	21 secs	1 sec
Terminal Recovery (TR)	154 secs	2 secs
Convert to Primary (C2P)	114 secs	5 secs
Open new Primary (OnP)	98 secs	28 secs
Open PDB and Start Service (OPDB)	146 secs	16 secs
Total App Downtime	**533 secs or 8min 53 secs**	**52 secs (90% drop)**

The "Tuned" timings were achieved by implementing the following MAA recommended practices:

- Evaluate Data Guard Fast-Start Failover and test with different `FastStartFailoverThreshold` settings
- Use Bigfile Tablespace
- Oracle Data Guard Configuration Best Practices
- Role Transition, Assessment, and Tuning

Customer Examples

Real-world Data Guard role transition duration observations from Oracle customers are shown in the following table.

Primary and Data Guard Configuration	Observed RTO
Single instance database failover in Database Cloud (DBCS) with heavy OLTP workload. Data Guard threshold is 5 seconds.	20 secs
Large Commercial Bank POC Results with 4 node RAC with heavy OLTP workload	51 secs (unplanned DR) 82 secs (planned DR)

Primary and Data Guard Configuration	Observed RTO
ExaDB-D 2-node RAC MAA testing with heavy OLTP workload	78 secs
ADB-D 2-node RAC MAA testing (25 PDBs, 250 services) with heavy OLTP workload	104 secs
ADB-D 2-node RAC MAA testing (50 PDBs, 600 services) with heavy OLTP workload	180 secs
ADB-D 2-node RAC MAA testing (4 CDBs, 100 PDBs total, 500 services) with heavy OLTP workload	164 secs
Oracle SaaS Fleet (thousands) of 12-node RAC, 400 GB SGA, 4000+ data files (note: reducing number of data files to hundreds can reduce downtime by minutes)	< 6 mins
Third Party SaaS Fleet (thousands) of 7-12 node RACs with quarterly site switch	< 5 mins

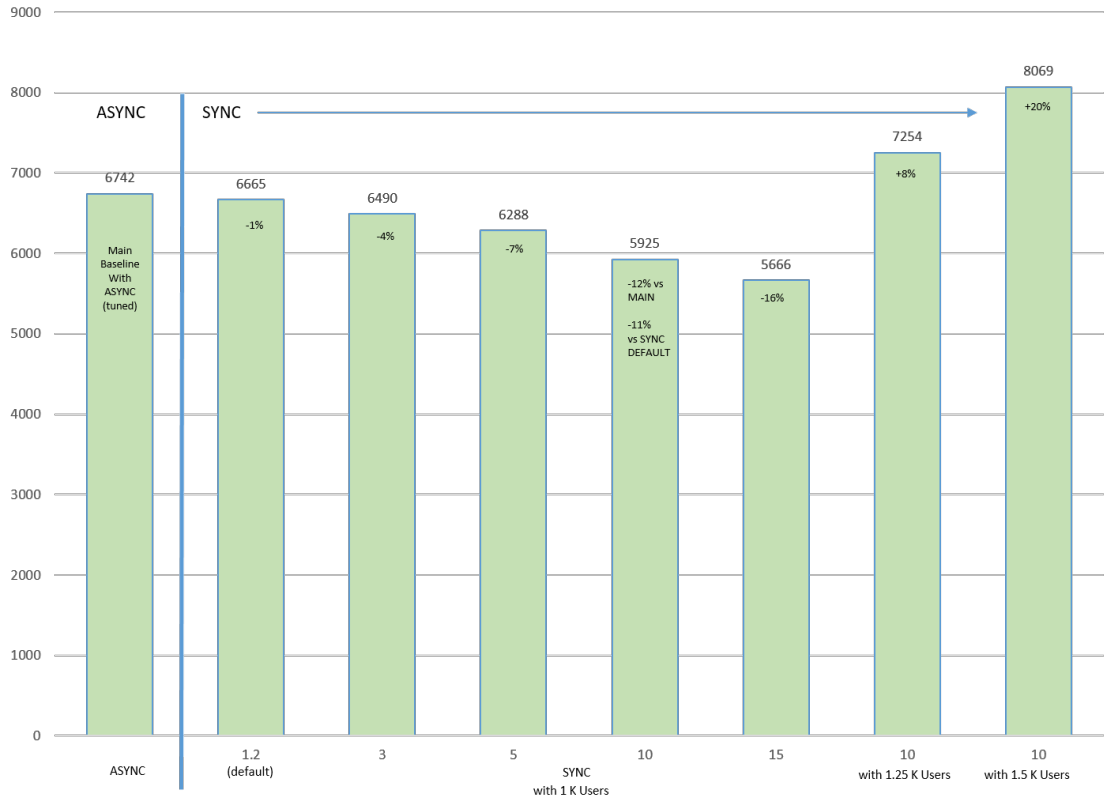
Application Throughput and Response Time Impact with Data Guard

Application throughput and response time impact is near zero when you enable Data Guard Max Performance protection mode or ASYNC transport. Throughput and application response time is typically not impacted at all in those cases.

With Data Guard Max Availability or Max Protection mode or SYNC transport, the application performance impact varies, which is why application performance testing is always recommended before you enable SYNC transport. With a tuned network and low round-trip latency (RTT), the impact can also be negligible, even though every log commit has to be acknowledged to every available SYNC standby database in parallel to preserve a zero data loss solution.

Here's an example of the application throughput impact but application impact varies based on workload:

Figure 16-3 Application impact with MTU=9000



Notice the lower network RTT latency (x axis), the application (TPS or y axis) throughput reduces.

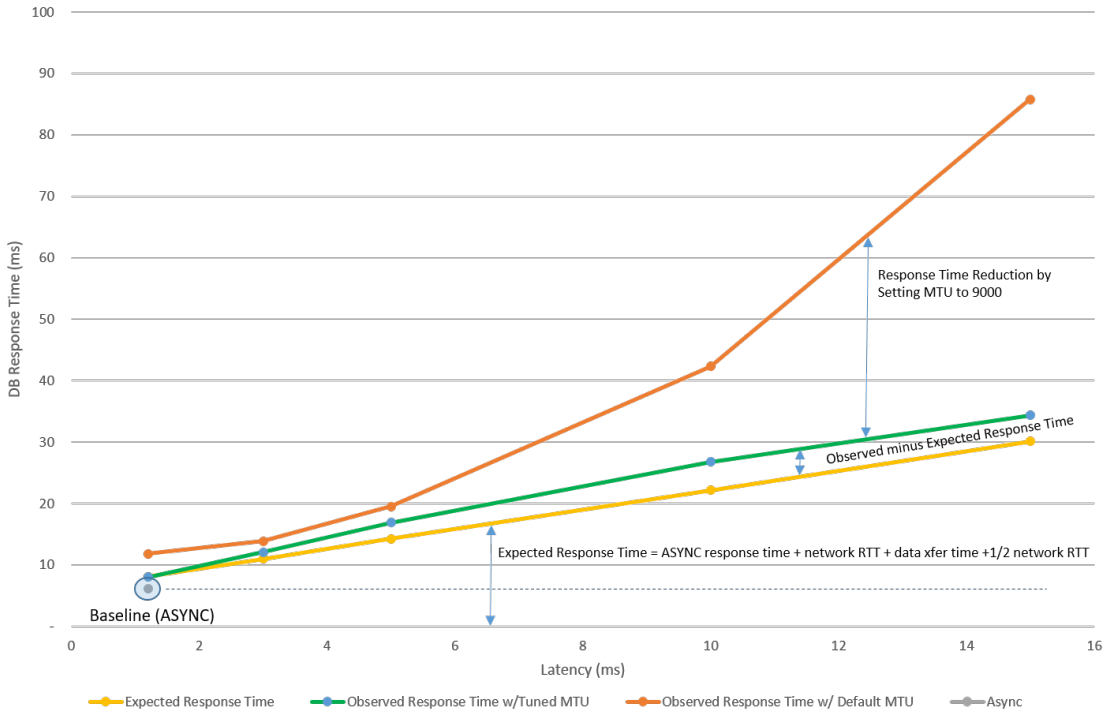
Note that in this network environment we observed that increasing MTU from 1500 (default) to 9000 (for example, jumbo frames) helped significantly since log message size increased significantly with SYNC. With the larger MTU size, the number of network packets per redo send request are reduced.

See Assessing and Optimizing Network Performance for details about tuning the network including the socket buffer size and MTU.

Even when throughput decreases significantly with higher RTT latency, you can increase TPS if your application can increase the concurrency. In the above chart, the last 2 columns increased the workload concurrency by adding more users.

Application response time with SYNC transport can also increase, but will vary based on each application workload and network tuning. With SYNC transport, all log writes have to wait for standby SYNC acknowledgment. This additional wait result in more foregrounds waiting for commit acknowledgment. Because commits have to be acknowledged by the standby database and more foregrounds are waiting for commits, the average log write size increases which affects the redo/data transfer time, as shown in the following chart.

Figure 16-4 Database response time (ms) vs latency (ms) for tuned and default MTU



In this example, we observed from AWR reports that average redo write size increased significantly, and tuning MTU reduced the response time impact. See Assessing and Optimizing Network Performance on tuning network including the socket buffer size and MTU.

After tuning the network, the response time impact was very predictable and low. Note that response impact varies per application workload.

To get the best application performance with Data Guard, use the following practices:

- Tune the application without Data Guard first and you should observe similar performance for ASYNC transport
- Implement Oracle Data Guard Configuration Best Practices
- Use Redo Transport Troubleshooting and Tuning methods
- Tune the network to improve application performance with SYNC. See Assessing and Optimizing Network Performance
- Application workload specific changes that can help increase throughput for SYNC Transport are:
  - Evaluate adding more concurrency or users to increase throughput.
  - For non-critical workloads within certain sessions that do not require zero data loss, evaluate advanced `COMMIT_WRITE` attribute to `NOWAIT`.

    In this case, you can commit before receiving the acknowledgment. Redo is still sent to persistent redo logs but is done asynchronously. Recovery is guaranteed for all persistent committed transactions in the redo that is applied. See COMMIT_WRITE in *Oracle Database Reference*.

17

Monitor an Oracle Data Guard Configuration

Use the following Oracle MAA best practice recommendations to monitor an Oracle Data Guard configuration.

Monitoring Oracle Data Guard Configuration Health Using the Broker

The Oracle data Guard broker issues a health check once a minute and updates the configuration status. To force a health check to occur immediately, run the command `show configuration verbose`.

On a primary database, the health check determines if the following conditions are met.

- Database is in the state specified by the user, as recorded in the broker configuration file
- Database is in the correct data protection mode
- Database is using a server parameter file(`SPFILE`)
- Database is in the `ARCHIVELOG` mode
- Redo transport services do not have any errors
- Database settings match those specified by the broker configurable properties
- Redo transport settings match those specified by the redo transport-related properties of the standby databases
- Current data protection level is consistent with configured data protection mode
- Primary database is able to resolve all gaps for all standby databases

On a standby database, the health check determines whether the following conditions are met.

- Database is in the state specified by the user, as recorded in the broker configuration file
- Database is using a server parameter file (`SPFILE`)
- Database settings match those specified by the broker configurable properties
- Primary and target standby databases are synchronized or within lag limits if fast-start failover is enabled

To identify any warnings on the overall configuration, show the status using the `SHOW CONFIGURATION` command.

```
DGMGRL> show configuration;

Configuration - dg

  Protection Mode: MaxPerformance
  Members:
  tin - Primary database
    can - Physical standby database
```

```
Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS   (status updated 18 seconds ago)
```

If the configuration status is `SUCCESS`, everything in the broker configuration is working properly.

However, if you see a status of `WARNING` or `ERROR`, then something is wrong in the configuration. Additional error messages will accompany the `WARNING` or `ERROR` status that should be used to identify current issues.

The next step is to examine each database in the configuration to narrow down what the specific error is related to.

To identify the warnings on the primary database, get its status using the `SHOW DATABASE` command.

```
DGMGRL> show database tin

Database - tin

  Role:               PRIMARY
  Intended State:     TRANSPORT-ON
  Instance(s):
    tin1
    tin2

Database Status:
SUCCESS
```

If the database status is `SUCCESS` then the database is working properly.

However, if you see a status of `WARNING` or `ERROR`, then something is wrong in the database. Additional error messages will accompany the `WARNING` or `ERROR` status that should be used to identify current issues.

Repeat the same `SHOW DATABASE` command on the standby database and assess any error messages.

In addition to the above commands, the broker features a `VALIDATE DATABASE` command.

```
DGMGRL> validate database tin

  Database Role:    Primary database
  Ready for Switchover:  Yes

DGMGRL> validate database can;

  Database Role:     Physical standby database
  Primary Database:  tin

  Ready for Switchover:  No
  Ready for Failover:    Yes (Primary Running)

  Capacity Information:
    Database  Instances        Threads
```

```
    tin       2               2
    can       1               2
    Warning: the target standby has fewer instances than the
    primary database, this may impact application performance

  Standby Apply-Related Information:
    Apply State:      Not Running
    Apply Lag:        Unknown
    Apply Delay:      0 minutes
```

The `VALIDATE DATABASE` does not provide a `SUCCESS` or `WARNING` status and must be examined to determine if any action needs to be taken.

It is recommended that you run the `VALIDATE DATABASE` command after creating the broker configuration, and before and after any role transition operation.

The `VALIDATE DATABASE` command performs the following checks.

- Whether there is missing redo data on a standby database
- Whether flashback is enabled
- The number of temporary tablespace files configured
- Whether an online data file move is in progress
- Whether online redo logs are cleared for a physical standby database
- Whether standby redo logs are cleared for a primary database
- The online log file configuration
- The standby log file configuration
- Apply-related property settings
- Transport-related property settings
- Whether there are any errors in the Automatic Diagnostic Repository (for example, control file corruptions, system data file problems, user data file problems)

Detecting Transport or Apply Lag Using the Oracle Data Guard Broker

Given enough resources, in particular network bandwidth, an Oracle Data Guard standby can maintain pace with very high workloads. In cases where resources are constrained, the standby can begin to fall behind, resulting in a transport or apply lag.

A **transport lag** is the amount of data, measured in time, that the standby has not received from the primary.

An **apply lag** is the difference, in elapsed time, between when the last applied change became visible on the standby and when that same change was first visible on the primary.

When using the Data Guard broker, the transport or apply lag can be viewed by using the `SHOW DATABASE` command and referencing the standby database, as shown here.

```
DGMGRL> show database orclsb

Database - orclsb

  Role:               PHYSICAL STANDBY
  Intended State:     APPLY-ON
```

```
  Transport Lag:      0 seconds (computed 0 seconds ago)
  Apply Lag:          0 seconds (computed 1 second ago)
  Average Apply Rate: 792.00 KByte/s
  Real Time Query:    ON
  Instance(s):
    orclsb1 (apply instance)
    orclsb2

Database Status:
SUCCESS
```

The broker `TransportDisconnectedThreshold` database property (default of 0 in Oracle Database 11.2, and 30 seconds for Oracle Database 12.1 and later releases) can be used to generate a warning status for a standby when the last communication from the primary database exceeds the value specified by the property. The property value is expressed in seconds.

The following is an example of the warning when a disconnection has occurred.

```
DGMGRL> show database orclsb;

Database - orclsb

  Role:               PHYSICAL STANDBY
  Intended State:     APPLY-ON
  Transport Lag:      0 seconds (computed 981 seconds ago)
  Apply Lag:          0 seconds (computed 981 seconds ago)
  Average Apply Rate: 12.00 KByte/s
  Real Time Query:    OFF
  Instance(s):
    orclsb1 (apply instance)
    orclsb2

  Database Warning(s):
    ORA-16857: member disconnected from redo source for longer than specified
threshold
```

The broker also has the following configurable database properties that you can use to generate warnings when a transport or apply lag exceed a user defined value.

- The `ApplyLagThreshold` property generates a warning status for a logical or physical standby when the database's apply lag exceeds the value specified by the property.

  The property value is expressed in seconds. A value of 0 seconds results in no warnings being generated when an apply lag exists. As a best practice, Oracle recommends setting `ApplyLagThreshold` to at least 15 minutes.

- The `TransportLagThreshold` property can be used to generate a warning status for a logical, physical, or snapshot standby when the database's transport lag exceeds the value specified by the property.

  The property value is expressed in seconds. A value of 0 seconds results in no warnings being generated when a transport lag exists. As a best practice, Oracle recommends setting `TransportLagThreshold` to at least 15 minutes.

Monitoring Oracle Data Guard Configuration Health Using SQL

You can use the queries in the following tables to assess the overall Data Guard configuration health on the primary database and the standby database.

Table 17-1 Primary Database Queries

Goal	Query	Expected Results
Check if any remote standby archive destination is getting errors Check if all remote standby archive destinations is enabled or `VALID`	`select sysdate,status,error from gv$archive_dest_status where type='PHYSICAL' and status!='VALID' or error is not null;`	Good health = no rows returned If the query returns rows, then raise an alert with the returned data.
Check if any `NOLOGGING` activity occurred on the primary database in the last day	`select file#, name, unrecoverable_change#, unrecoverable_time from v$datafile where unrecoverable_time > (sysdate - 1);`	Good health = no rows returned If the query returns rows, then the standby database is vulnerable, and the files listed in the output must be refreshed on the standby.
Detect gaps on the standby database	`select sysdate,database_mode,recovery_mode, gap_status from v$archive_dest_status where type='PHYSICAL' and gap_status !='NO GAP';`	Good health = no rows returned If the query returns rows, then there's an existing gap between the primary and the standby database, and you must run the same query on the standby database. If the output from the primary and standby is identical, then no action is required. If the output on the standby does not match the output from the primary, then the datafile on the standby should be refreshed.
Assess whether any severe Data Guard event occurred in the last day	`select * from v$dataguard_status where severity in ('Error','Fatal') and timestamp > (sysdate -1);`	Good health = no rows returned If the query returns rows, then raise an alert with the returned output.

Table 17-1 (Cont.) Primary Database Queries

Goal	Query	Expected Results
FOR SYNC ENVIRONMENTS ONLY: Assess if running in Maximum Availability mode and configuration is in sync	`select sysdate,protection_mode, synchronized, synchronization_status from v$archive_dest_status where type='PHYSICAL' and synchronization_status !='OK';`	Good health = no rows returned If the query returns rows, then raise an alert with the returned output.

Table 17-2 Physical Standby Database Queries

Goal	Query	Expected Results
Determine if there is a transport lag	`select name,value,time_computed,datum_time from v$dataguard_stats where name='transport lag' and value > '+00 00:01:00';`	Good health = no rows returned If no rows are returned, then this implies that there is no transport lag
Determine if there is an apply lag	`select name,value,time_computed,datum_time from v$dataguard_stats where name='apply lag' and value > '+00 00:01:00';`	Good health = no rows returned If no rows are returned, then this implies that there is no apply lag
Standby data file check (offline files or files that are not accessible)	`select * from v$datafile_header where status ='OFFLINE' or ERROR is not null;`	Good health = no rows returned Any rows returned list the files that have I/O or recovery issues
Verify that the Media Recovery Process is currently running	`select * from v$managed_standby where process like 'MRP%';`	Good health = rows returned If no rows are returned, then the MRP process is not running

Table 17-2 (Cont.) Physical Standby Database Queries

Goal	Query	Expected Results
Assess whether any severe Data Guard event occurred in the last day	`select * from v$dataguard_status where severity in ('Error','Fatal') and timestamp > (sysdate -1);`	Good health = no rows returned If the query returns rows, then raise an alert with the returned output

Oracle Data Guard Broker Diagnostic Information

The Oracle Data Guard broker provides information about its activities in several forms.

- Database status information
- Oracle alert log files

  The broker records key information in the alert log file for each instance of each database in a broker configuration.
- Oracle Data Guard broker log files

  For each instance of each database in a broker configuration, the broker DMON process records important behavior and status information in a broker log file, which are useful for diagnosing Oracle Data Guard failures. The Set the `TraceLevel` configuration property to specify the level of diagnostic information reported in the broker log files. The broker log file is created in the same directory as the alert log and is named `drc<$ORACLE_SID>.log`.
- Oracle Data Guard command line (DGMGRL) `logfile` option

  If the DGMGRL command-line interface was started with the `-logfile` optional parameter, then the resulting log file may contain a useful record of past operations and error conditions.

Detecting and Monitoring Data Corruption

If corrupt data is written to disk, or if a component failure causes good data to become corrupt after it is written, then it is critical that you detect the corrupted blocks as soon as possible.

To monitor the database for errors and alerts:

- Query the `V$DATABASE_BLOCK_CORRUPTION` view that is automatically updated when block corruption is detected or repaired.
- Configure Data Recovery Advisor to automatically diagnose data failures, determine and present appropriate repair options, and perform repair operations at your request.

  Note that Data Recovery Advisor integrates with the Oracle Enterprise Manager Support Workbench (Support Workbench), the Health Monitor, and RMAN.
- Use Data Guard to detect physical corruptions and to detect lost writes.

  Data Guard can detect physical corruptions when the apply process stops due to a corrupted block in the redo steam or when it detects a lost write.

Use Enterprise Manager to manage and monitor your Data Guard configuration.

By taking advantage of Automatic Block Media Recovery, a corrupt block found on either a primary database or a physical standby database can be fixed automatically when the Active Data Guard option is used.

- Use SQL*Plus to detect data file corruptions and inter-block corruptions.

  Run this SQL*Plus statement:

  ```
  sqlplus> ANALYZE TABLE table_name VALIDATE STRUCTURE CASCADE;
  ```

  After finding the corruptions, the table can be re-created or another action can be taken.

- An Recovery Manager (RMAN) backup and recovery strategy can detect physical block corruptions.
  A more intensive RMAN check using the following command can detect logical block corruptions.

  ```
  RMAN> BACKUP VALIDATE CHECK LOGICAL;
  ```

Part V

MAA Platinum and Oracle GoldenGate Best Practices

18

MAA Platinum Reference Architecture Overview

MAA Platinum or Never-Down Architecture, delivers near-zero Recovery Time Objective (RTO, or downtime incurred during an outage) and potentially zero or near zero Recover Point Objective (RPO, or data loss potential).

The MAA Platinum reference architecture ensures:

- RTO = zero or near-zero for all local failures using the Oracle Exadata Database Machine platform with its inherent Oracle RAC, full-stack redundancy, and failover capabilities
- RTO = zero or near-zero for disasters, such as database, cluster, or site failures, achieved by redirecting the application to an active Oracle GoldenGate source or target
- Zero downtime maintenance for software and hardware updates using Oracle RAC and Exadata Database Machine platform
- Zero downtime database upgrade or application upgrade by redirecting the application to an upgraded Oracle GoldenGate source or target database
- RPO = zero or near-zero data loss, depending on the Oracle Data Guard protection mode setting, which dictates the redo transport (SYNC, FAR SYNC, or ASYNC)
- Fast re-synchronization and zero or near-zero RPO between Oracle GoldenGate source and target databases after a disaster.

  After any database failure, automatic failover to its standby database occurs automatically. Subsequently, automatic re-synchronization between Oracle GoldenGate source and target databases will resume. For SYNC transport, this leads to eventual zero data loss.

Table 18-1 MAA Platinum Outage Matrix

Event	RTO/RPO Service Level Objective[1]
Unplanned Outage	
Recoverable node or instance failure	Zero or single digit seconds[2,3]
Disasters including corruptions and site failures	Zero[3]
Planned Maintenance	
Most common software and hardware updates	Zero[2]
Major database upgrade or application upgrade	Zero[3]

[1]RPO=0 unless explicitly specified

[2]To achieve zero downtime or lowest impact for online processing, apply MAA application high availability best practices (also known as The Checklist). For long running transactions, such as batch operations, it's recommended that you defer them outside the planned maintenance window.

[3]Application failover is customized or managed with Global Data Services.

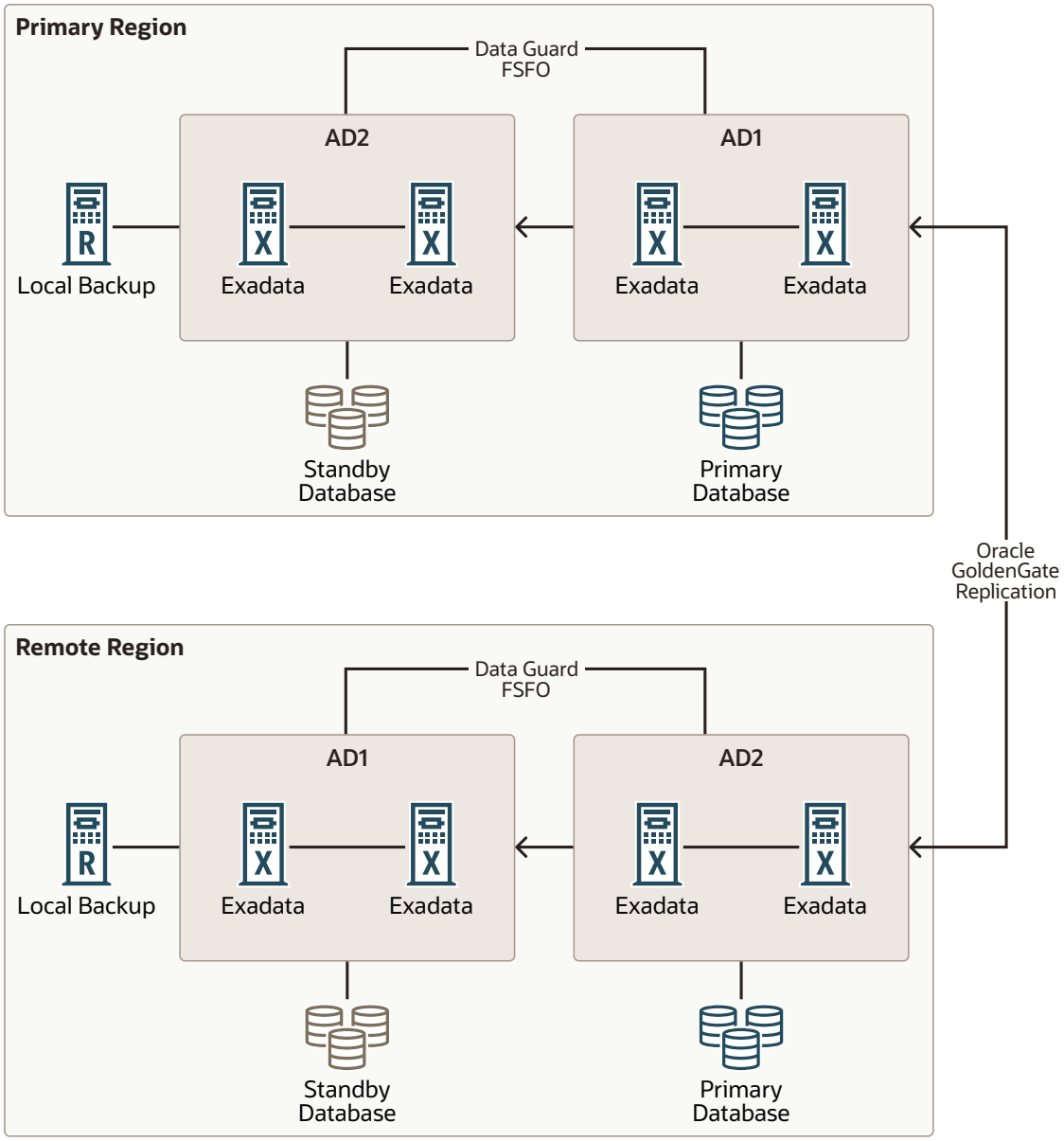
Enabling features of the Platinum MAA solution include:

- **Oracle Real Application Clusters** with recommended **Exadata** Database Machine Platform, Exadata Database Service on Dedicated Infrastructure (ExaDB-D), or Oracle Exadata Database Service on Cloud@Customer (ExaDB-C@C) for the source and target databases
- **Oracle Active Data Guard** with Fast-Start Failover to bound data loss and automatically fail over to standby in case of database, cluster, or data center failures. The standby databases are typically in separate Fault Domains (FDs) with separate power supplies, separate data centers, or Availability Domains (ADs) with independent power and network. The standbys can also reside across regions with typically the greatest fault isolation.
- **Oracle GoldenGate** enables two active read-write database systems that can be leveraged for applications to fail over immediately after database, cluster, site failure, or planned outages such as database or application upgrades. The source and target databases on which Oracle GoldenGate replication is occurring can reside in the same region, across ADs, or across regions.

MAA Platinum architecture is illustrated in the image below, where two "active read-write" primary databases, or source or target databases, reside in separate regions. Oracle GoldenGate replication occurs between the source and target databases between primary and remote regions. Each primary database is protected by a standby database in another AD within the same region.

With Data Guard Fast-Start Failover (FSFO), the standby database becomes the new primary database automatically after primary database, cluster, or AD failure. With MAA Oracle GoldenGate configuration best practices implemented, replication resumes automatically between the source and target databases after any Data Guard role transition. Each database resides on an Exadata platform with its inherent built-in Real Application Cluster, system and storage redundancy, and low brownout failover capabilities.

Figure 18-1 MAA Platinum Reference Architecture



MAA Platinum Architecture Variants

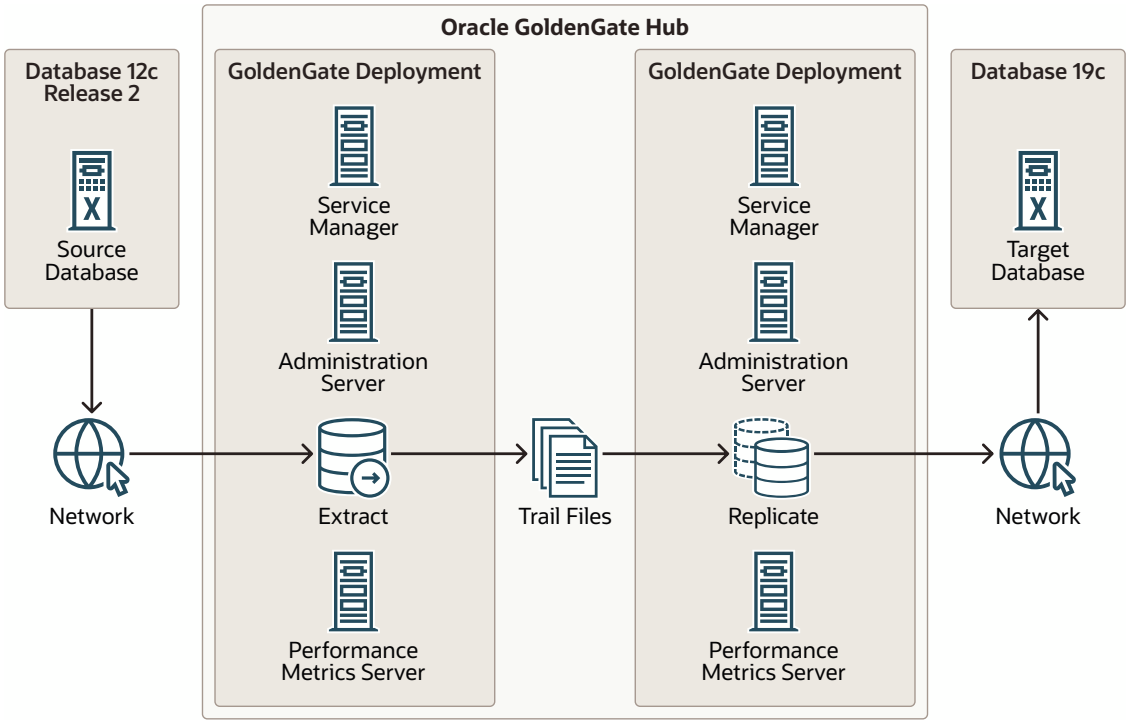
When setting up MAA Platinum architecture, the administrator can decide between setting up Oracle GoldenGate on each potential source or target database, or creating an Oracle GoldenGate hub independent from the database servers.

The MAA Oracle GoldenGate hub, shown in the following image, provides the following advantages:

- Offloads Oracle GoldenGate software installation, configuration, and life cycle management from source and target Exadata database systems.
- Reduces Oracle GoldenGate resource impact on the source and target database systems.

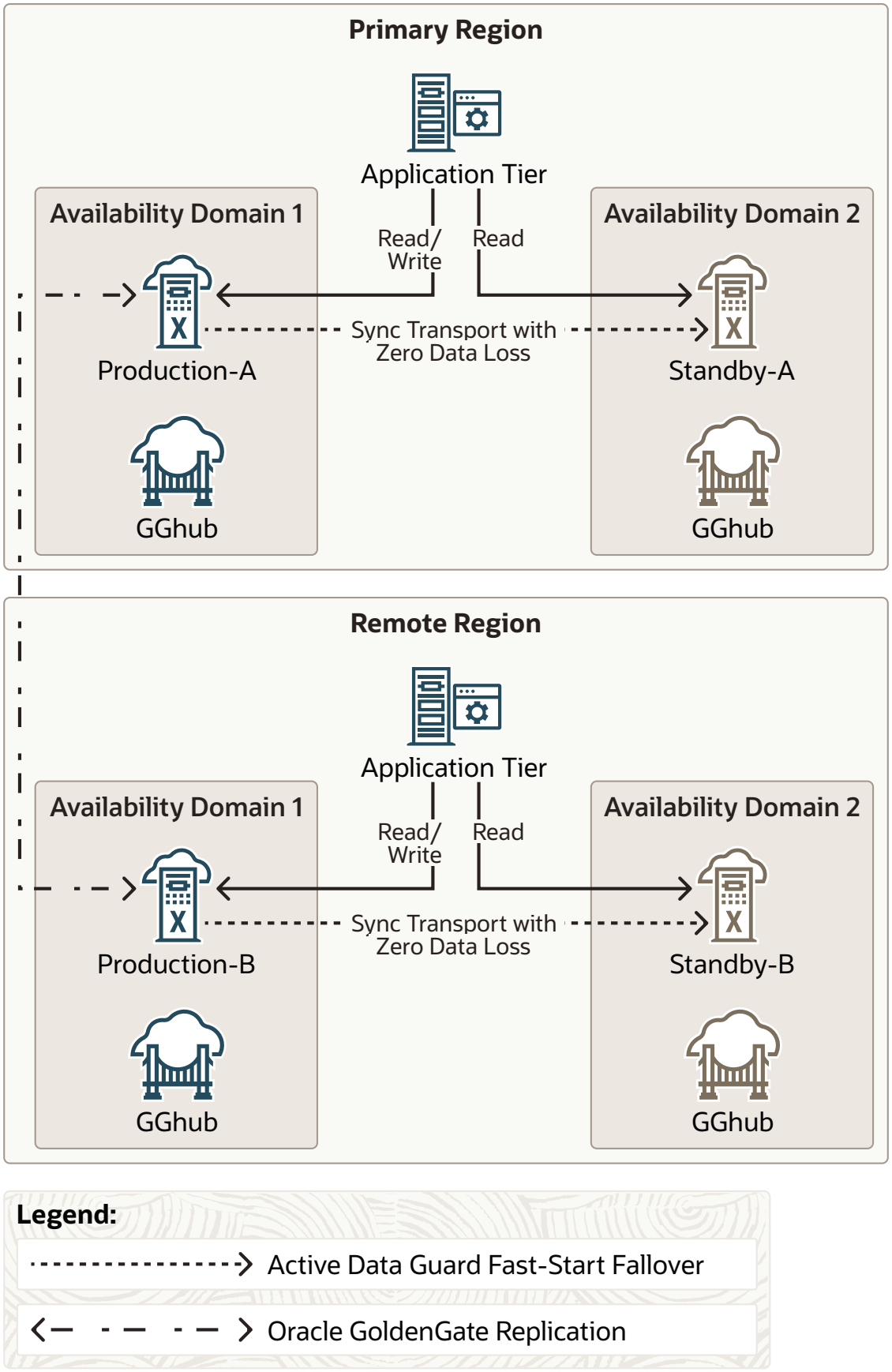
- Provides high availability by configuring a 2-node cluster server for fast and simple failover, and disaster recovery by leveraging ACFS replication to another identical GoldenGate hub server on a separate 2-node cluster server.
- Consolidates Oracle GoldenGate configurations and software deployment for multiple independent MAA Platinum or Oracle GoldenGate architectures.

Figure 18-2 MAA Oracle GoldenGate Hub



An example of MAA Oracle GoldenGate hub with MAA Platinum architecture is shown in the image below. Each hub is a 2-node cluster providing local high availability, and for additional protection uses ACFS replication to another hub, typically deployed across Availability Domains (ADs) or across regions.

Figure 18-3 MAA Platinum with Oracle GoldenGate Hub



How to Implement the MAA Platinum Solution

To achieve an MAA Platinum solution, review and leverage the technical papers and documentation referenced in the following steps.

1. Review Oracle MAA Platinum Tier for Oracle Exadata to understand MAA Platinum benefits and use cases.
2. Decide between implementing the MAA Oracle GoldenGate hub solution, or setting up Oracle GoldenGate directly on database servers.
   - Option 1: (RECOMMENDED) Configure MAA Oracle GoldenGate Hub
     - See Cloud: Configuring Oracle GoldenGate Hub for MAA Platinum for the Oracle Cloud configuration, or
     - On-Premises: Configuring Oracle GoldenGate Hub for on-premises configuration.
   - Option 2: Configure Oracle GoldenGate on the Exadata Database servers
     - For Oracle Cloud Service, see
       a. Cloud: Oracle GoldenGate Microservices Architecture on Oracle Exadata Database Service Configuration Best Practices and
       b. Cloud MAA Platinum: Oracle GoldenGate Microservices Architecture Integrated with Active Data Guard
     - For on-premises systems, see
       a. On-Premises: Oracle GoldenGate Microservices Architecture with Oracle Real Application Clusters Configuration Best Practices and
       b. On-Premises MAA Platinum: Oracle GoldenGate Microservices Architecture Integrated with Active Data Guard
3. Configure Bidirectional Replication and Automatic Conflict Detection and Resolution. See Oracle Cloud Infrastructure GoldenGate documentation or the latest Oracle GoldenGate documentation.
4. Configure application failover options such as
   - Global Data Services (see Global Data Services documentation) and
   - MAA application high availability configuration (also known as the "Checklist") at Configuring Continuous Availability for Applications.

19

Overview of Oracle GoldenGate Best Practices

Configure Oracle GoldenGate using Oracle MAA best practices to get the highest availability and performance out of your Oracle GoldenGate deployment.

Oracle GoldenGate provides the following benefits:

- Uni-directional or bi-directional replication, allowing reads and updates in any replicated database.
- Data movement is in real-time, reducing latency.
- Replicated databases can run on different hardware platforms, database versions, and different database or application configurations, allowing for online migration. This flexibility also allows online database and application upgrades.
- Source and target replicated databases are online, so zero downtime switch over of applications, during outages and planned maintenance activities is possible. Note, the application switchover must be customized, rather than using a built-in feature, such as Transparent Application Continuity.

The following table highlights various Oracle GoldenGate configuration best practices and MAA Platinum best practices.

Table 19-1 Oracle GoldenGate Use Cases and Best Practices

Use Case	Oracle GoldenGate Best Practices
Database migration to Oracle Cloud or Exadata Platform	Zero Downtime Migration (ZDM) with GoldenGate (logical migration) Oracle Zero Downtime Migration – Logical Migration Performance Guidelines
Database migration requiring minimal or zero downtime Database migration involving cross platform or different database versions	Oracle Database Migration with an Oracle GoldenGate Hub Configuration
Deploy Oracle GoldenGate off of the database server in a Hub configuration	Cloud: Configuring Oracle GoldenGate Hub for MAA Platinum On-Premises: Configuring Oracle GoldenGate Hub
Install and configure Oracle GoldenGate directly on the Oracle RAC database server or Exadata database system	Cloud: Oracle GoldenGate Microservices Architecture on Oracle Exadata Database Service Configuration Best Practices On-Premises: Oracle GoldenGate Microservices Architecture with Oracle Real Application Clusters Configuration Best Practices

Table 19-1 (Cont.) Oracle GoldenGate Use Cases and Best Practices

Use Case	Oracle GoldenGate Best Practices
Implement MAA Platinum or install and configure Oracle GoldenGate directly on Oracle RAC database servers with Oracle Active Data Guard	Cloud: Oracle GoldenGate Microservices Architecture on Oracle Exadata Database Service Configuration Best Practices and Cloud MAA Platinum: Oracle GoldenGate Microservices Architecture Integrated with Active Data Guard On-Premises: Oracle GoldenGate Microservices Architecture with Oracle Real Application Clusters Configuration Best Practices and On-Premises MAA Platinum: Oracle GoldenGate Microservices Architecture Integrated with Active Data Guard
Application failover options for Oracle GoldenGate	Global Data Services Concepts and Administration Guide and Configuring Continuous Availability for Applications
Oracle GoldenGate bidirectional replication and automatic conflict detection and resolution	For Oracle Cloud Service, Oracle Cloud Infrastructure GoldenGate documentation For on-premises, the latest Oracle GoldenGate documentation

Also see Oracle GoldenGate documentation at: https://docs.oracle.com/en/middleware/goldengate/core/21.1/

Overview of Oracle GoldenGate and Supporting Technologies

The technologies that are required to replicate data between databases are Oracle GoldenGate and supporting technologies (such as Oracle Grid Infrastructure Agents, ACFS or DBFS, or GoldenGate hub) to ensure that replication will resume after various failures. A brief overview of Oracle GoldenGate and supporting technologies are described here.

Oracle GoldenGate

Oracle GoldenGate provides real-time, log-based change data capture and delivery between homogenous and heterogeneous systems. This technology lets you construct a cost-effective and low-impact real-time data integration and continuous availability solution.

Oracle GoldenGate replicates data from committed transactions with transaction integrity and minimal overhead on your existing infrastructure. The architecture supports multiple data replication topologies, such as one-to-many, many-to-many, cascading, and bidirectional. Its wide variety of use cases includes real-time business intelligence; query offloading; zero-downtime upgrades and migrations; and active-active databases for data distribution, data synchronization, and high availability.

Oracle GoldenGate Microservices Architecture provides REST-enabled services. The REST-enabled services provide remote configuration, administration, and monitoring through HTML5 web pages, command line interfaces, and APIs.

Recommended Oracle GoldenGate 21c (and higher releases) introduces unified build support, so that a single software installation supports capturing and applying replicated data to multiple major Oracle Database versions (11g Release 2 to 21c). This is possible because an Oracle GoldenGate installation includes the required Oracle Database client libraries without requiring a separate database `ORACLE_HOME` installation.

Oracle Grid Infrastructure Agents

Oracle Grid Infrastructure Agents (XAG) are Oracle Grid Infrastructure components that provide the high availability (HA) framework to application resources and resource types managed through the agent management interface, AGCTL. This framework provides a complete, ready-to-use solution that contains pre-defined Oracle Grid Infrastructure resource configurations and agents to integrate applications for complete application HA.

The Oracle Grid Infrastructure Agents provide pre-defined Oracle Clusterware resources for Oracle GoldenGate, Siebel, Oracle PeopleSoft, JD Edwards, and Oracle WebLogic Server, as well as Apache and MySQL applications. Using the agent for Oracle GoldenGate simplifies the creation of dependencies on the source and target databases, the application VIP, and the file system (ACFS or DBFS) mount point. The agent command line utility (AGCTL) is used to start and stop Oracle GoldenGate, and can also be used to relocate Oracle GoldenGate between the nodes in the cluster.

Oracle Database File System (DBFS)

Oracle DBFS can be used to store Oracle GoldenGate files.

The Oracle Database File System (DBFS) creates a file system interface to files stored in the database. DBFS is similar to NFS in that it provides a shared network file system that looks like a local file system. Because the data is stored in the database, the file system inherits all the high availability and disaster recovery capabilities provided by Oracle Database.

With DBFS, the server is the Oracle Database. Files are stored as SecureFiles LOBs. PL/SQL procedures implement file system access primitives such as create, open, read, write, and list directory. The implementation of the file system in the database is called the DBFS SecureFiles Store. The DBFS SecureFiles Store allows users to create file systems that can be mounted by clients. Each file system has its own dedicated tables that hold the file system content.

Oracle Advanced Cluster File System (ACFS)

Oracle ACFS can be used to store Oracle GoldenGate files.

Oracle Advanced Cluster File System (Oracle ACFS) is a multi-platform, scalable file system, and storage management technology that extends Oracle Automatic Storage Management (Oracle ASM) functionality to support all customer files.

Oracle ACFS leverages Oracle Clusterware for cluster membership state transitions and resource-based high availability. Oracle ACFS is bundled into the Oracle Grid Infrastructure (GI) allowing for integrated optimized management of databases, resources, volumes, and file systems.

20

Cloud: Configuring Oracle GoldenGate Hub for MAA Platinum

Configure and deploy MAA Oracle GoldenGate Hub architecture on Oracle Cloud using the provided planning considerations, tasks, management, and troubleshooting information.

See the following topics:

Overview of MAA GoldenGate Hub

To achieve the highest levels of availability, resulting in zero or near-zero downtime for both unplanned outages and planned maintenance activities, you can use the combination of Oracle Real Application Clusters (Oracle RAC), Oracle Active Data Guard, and Oracle GoldenGate.

This architecture, typically referred as MAA Platinum, or Never Down Architecture, delivers near zero Recovery Time Objective (RTO--downtime incurred during outage) and potentially zero or near zero Recovery Point Objective (RPO--data loss potential).

Traditionally, Oracle GoldenGate is installed and run locally on the database server that the GoldenGate processes connect to. When used with Oracle Grid Infrastructure Standalone Agent (XAG), Oracle GoldenGate processes can be configured to seamlessly relocate or failover between Oracle RAC nodes and follow Oracle Active Data Guard switchover and failovers.

Using MAA Oracle GoldenGate Hub (MAA GGHub) moves the GoldenGate software and processes off of the Exadata database servers, reducing complexity and system resource utilization. MAA GGHub centralizes Oracle GoldenGate management and offloads the majority of the Oracle GoldenGate processing and associated CPU and storage resource utilization from Exadata system resources. Connectivity between the GoldenGate processes and the databases they operate against is managed with Oracle Net Services.

To achieve an MAA Platinum solution in the Oracle Cloud, you follow these high level steps:

1. Review Oracle MAA Platinum Tier for Oracle Exadata to understand Platinum MAA benefits and use cases.
2. Deploy or migrate your database onto Exadata Cloud Service, Base Database Service, or Autonomous Database on Dedicated Infrastructure Service.
3. Add symmetric standby databases in the Oracle Cloud using Oracle Cloud Control Plan or Cloud automation.

4. Configure and deploy Oracle Data Guard Fast Start Failover using the Oracle MAA best practice recommendations in Configure Fast Start Failover. For Exadata Cloud Service and Base Database Service, this is manual step.
5. Set up MAA GGHub, which is detailed in the topics that follow.
6. Configure Bidirectional Replication and Automatic Conflict Detection and Resolution. See Oracle Cloud Infrastructure GoldenGate documentation for information.
7. Decide on Application Failover Options such as Global Data Services (see Introduction to Global Data Services), or use your own customized application failover.

Planning GGHub Placement in the Platinum MAA Architecture

Extreme availability that delivers zero downtime (RTO=0 or near zero) and zero or near zero data loss (RPO=0 or near zero) typically requires the following Platinum MAA architecture.

1. You have the source and target database in an Oracle GoldenGate architecture to allow your application to fail over immediately in the case of disaster (database, cluster, or site failure) or switch over in the case of a database or application upgrade. This architecture enables the potential RTO of zero or near zero for disaster scenarios and database and application upgrade maintenance.
2. Each source and target database is deployed in Exadata cloud systems so any local failures are tolerated or recovered almost instantly.
3. Each source and target database is configured with a standby database with Data Guard Fast-Start Failover so any failure of the database results in activating a new primary database in seconds to minutes. If SYNC transport is leveraged with Max Availability protection mode, zero data loss Data Guard failover is achieved.
4. Configured with GoldenGate replication using MAA GGhub between the source and target databases.
5. Configured so that any standby becoming a primary database due to Data Guard switchover or failover will automatically resynchronize with its target GoldenGate database. If zero data loss Data Guard switchover or failover occurs, GoldenGate resychronization ensures zero data loss across the distributed database environment.
6. Configured with GoldenGate Automatic Conflict Detection and Resolution, which is required after any Data Guard failover operation occurs.

Where to Place the MAA Primary GGHub and Standby GGHub

1. The GGHub Pair (Primary and Standby GGHub) must reside in the same OCI regions as each primary and standby database. For example:
   a. If the primary database is in AD1, Region A, and the standby database is in AD2, Region A, then the GGHub pair will reside in Region A.
   b. If the primary database is in Region A and the standby database is in Region B, then the GGHub pair will split between Region A and B. The primary, or active, GGHub must be co-located in the same OCI region as the target primary database.
2. Performance implications:
   a. Primary or active GGHub must reside in the same data center as the target database to ensure round trip latency of 4ms or less. (Replicat performance)
   b. Primary or active GGHub should be < 90 ms from the source database without incurring GoldenGate performance degradation. (Extract performance)

3. GoldenGate distribution path:
   a. A GoldenGate distribution path is required if the source and target GGHubs are in different regions and latency between the OCI regions is > 90 ms.
   b. In Oracle Cloud, when your Oracle GoldenGate source and target databases reside in the same region, or in different regions in the same country, you never need to set up a GoldenGate distribution path because the latency is always < 90 ms.

MAA GGHubs Placed in the Same OCI Region

In this scenario, the primary and standby database are located in the same OCI region, and so the primary (active) GGHub and the standby GGHub are also located in the same region.

The following architectural components comprise the GGHubs, as shown in the image below:

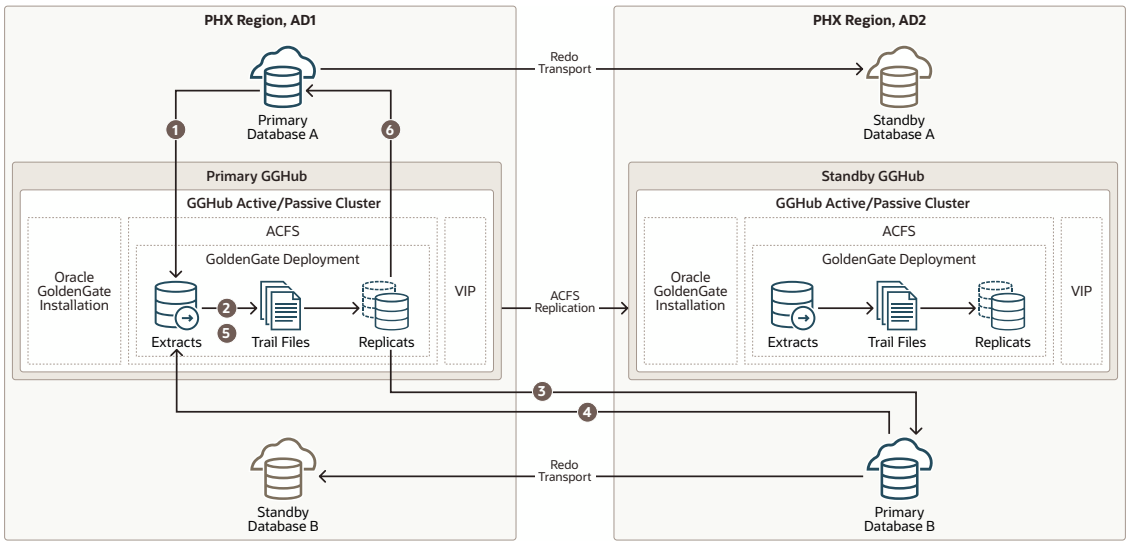
1. Primary database and associated standby database are configured with Oracle Active Data Guard Fast Start Failover (FSFO). FSFO can be configured with any Data Guard protection mode, with ASYNC or SYNC redo transport, depending on your maximum data loss tolerance.
2. Primary GGHub Active/Passive Cluster: Only one GGHub software deployment and configuration on the 2-node cluster. This cluster contains the 21c Oracle GoldenGate software deployment that can support Oracle Database 11g (11.2.0.4) and later releases.

   This GGHub can support many primary databases and encapsulates the GoldenGate processes. GoldenGate Extract mines transactions from the source database and GoldenGate Replicat applies the same changes to target database. GoldenGate trail and checkpoint files also reside in the GGhub ACFS file system.

   The HA failover solution is built in to the GGhub, which includes automatic failover to the passive node in the same cluster, and restarts GoldenGate processes and activity after a node failure.
3. Standby GGHub Active/Passive Cluster: A Symmetric standby GGhub is configured. ACFS replication is set up between the primary and standby GGHubs to preserve all GoldenGate files.

   Manual GGhub failover, which includes ACFS failover, can be performed in the rare case that you lose the entire primary GGhub.

Figure 20-1 Primary and Standby GGHubs in the Same OCI Region



The figure above depicts data replicated from Primary Database A to Primary Database B and Primary B back to Primary A with the following steps:

1. Primary Database A: Primary A's Logminer server sends redo changes to a Primary GGHub Extract process.
2. Primary GGHub: An Extract process writes changes to trail files.
3. Primary GGHub to Primary Database B: A Primary GGHub Replicat process applies those changes to the target database (Primary B).
4. Primary Database B: Primary B's Logminer server sends redo to a Primary GGHub Extract process.
5. Primary GGHub: A Primary GGHub Extract process writes changes to trail files.
6. Primary GGHub to Primary Database A: A Primary GGHub Replicat process applies those changes to the target database (Primary A).

Note that one GGHub can support multiple source and target databases, even when the source and target databases are different Oracle Database releases.

Table 20-1 Outage Scenarios, Repair, and Restoring Redundancy for GGHubs in the Same OCI Region

Outage Scenario	Application Availability and Repair	Restoring Redundancy and Pristine State
Primary Database A (or Database B) failure	**Impact**: Near-zero application downtime. GoldenGate replication resumes when a new primary database starts. 1. One primary database is still available. All activity is routed to the existing available primary database to achieve zero application downtime. Refer to Global Data Service Global Services Failover solution. For example, application services A-F are routed to Database A and application services G-J are routed to Database B. If Database A fails, all application services temporarily go to Database B. 2. The standby becomes the new primary automatically with Data Guard FSFO. Oracle GoldenGate replication resumes and the primary databases resynchronize. Data loss is bounded by the Data Guard protection level. If Maximum Availability or Maximum Protection is configured, zero data loss is achieved. All committed transactions are in one or both databases. Workload can be “rebalanced” when Primary Database A and Database B are available and in sync. For example, when Database A is up and running and in sync, services A-F can go back to Database A.	1. The old primary database is reinstated as the new standby database to restore redundancy. 2. Optionally performing a Data Guard switchover to switch back to the original configuration ensures that at least one primary database resides in an independent AD.
Primary or standby GGHub single node failure	**Impact**: No application impact. GoldenGate replication resumes automatically after a couple of minutes. No action is required. The HA failover solution built in to the GGHub includes automatic failover and restart of GoldenGate processes and activity. Replication activity is blocked until GoldenGate processes are active again. GoldenGate replication blackout could last a couple of minutes.	Once the node restarts, active/passive configuration is re-established.

Table 20-1 (Cont.) Outage Scenarios, Repair, and Restoring Redundancy for GGHubs in the Same OCI Region

Outage Scenario	Application Availability and Repair	Restoring Redundancy and Pristine State
Primary GGHub cluster crashes and is not recoverable	**Impact**: No application impact. GoldenGate replication resumes after restarting the existing GGHub or performing a manual GGHub failover operation. 1. If the GGHub cluster can be restarted, then that's the simplest solution. 2. If the primary GGHub is not recoverable, then perform a manual GGHub failover to the standby GGHub, which includes ACFS failover. This typically takes several minutes. 3. GoldenGate replication stops until the new primary GGhub is available, so performing step 1 or step 2 should take little time.	If the previous GGHub eventually restarts, ACFS replication resumes in the other direction automatically. If the GGHub cluster is lost or unrecoverable, you need to rebuild a new standby GGHub.
Standby GGHub cluster crashes and not recoverable	**Impact**: No application or replication impact. 1. If the GGHub cluster can be restarted, then that is the simplest solution, and ACFS replication can resume. 2. If the standby GGHub is not recoverable, you can rebuild a new standby GGHub.	N/A

Table 20-1 (Cont.) Outage Scenarios, Repair, and Restoring Redundancy for GGHubs in the Same OCI Region

Outage Scenario	Application Availability and Repair	Restoring Redundancy and Pristine State
Complete Data Center or Availability Domain (AD1 or AD2) failure	**Impact**: Near-zero application downtime. GoldenGate replication resumes when the new primary database starts. 1. One primary database is still available. All activity is routed to the existing available primary database to achieve zero application downtime. Refer to Global Data Service Global Services Failover solution. For example, application services A-F are routed to Database A and application services G-J are routed to Database B. If Database A fails, all services temporarily go to Database B. 2. If the primary GGHub is still functional, GoldenGate replication continues. If the primary GGHub is lost due to availability domain (AD) failure, then a manual GGhub failover is required. GoldenGate replication resumes and the primary databases resynchronize. Data loss is bounded by the Data Guard protection level. If Maximum Availability or Maximum Protection is configured, zero data loss is achieved. All committed transactions are in one or both databases. Workload can be rebalanced when Primary Database A and Database B are available and in sync. When Database A is up and running and in sync, services A-F can go back to Database A.	1. When the data center/AD returns, re-establish the configuration, such as reinstate standby. If the previous GGHub eventually restarts, ACFS replication resumes in the other direction automatically. 2. When possible, perform a Data Guard switchover (back) to get back to the original state where one primary database exists in each AD.

MAA GGHubs Placed in Different OCI Regions

In this scenario, the primary and standby database are located in different OCI regions, so the primary (active) GGHub is located in the same region as the primary database, and the standby GGHub is located in the same region as the standby database.

The following architectural components comprise the GGHubs, as shown in the image below:

1. The primary database and associated standby database are configured with Oracle Active Data Guard Fast Start Failover (FSFO). FSFO can be configured with any Data Guard protection mode, with ASYNC or SYNC redo transport, depending on your maximum data loss tolerance.

2. Primary GGHub Active/Passive Cluster: In this configuration, there's a 2-node cluster with two Oracle GoldenGate software configurations. Because the primary GGHub needs to be <= 4 ms from the target database, and the two regions (PHX and ASH) network latency > 5 ms, two GGHub configurations are created for each GGHub cluster. Essentially, a primary GGHub configuration will always be in the same region as the target database.

   GGHub is configured with the Oracle GoldenGate 21c software deployment that can support Oracle Database 11g and later releases. This GGHub can support many primary databases and encapsulates the GoldenGate processes. Extract mines transactions from the source database, and Replicat applies those changes to the target database. GoldenGate trail and checkpoint files also reside in the ACFS file system.

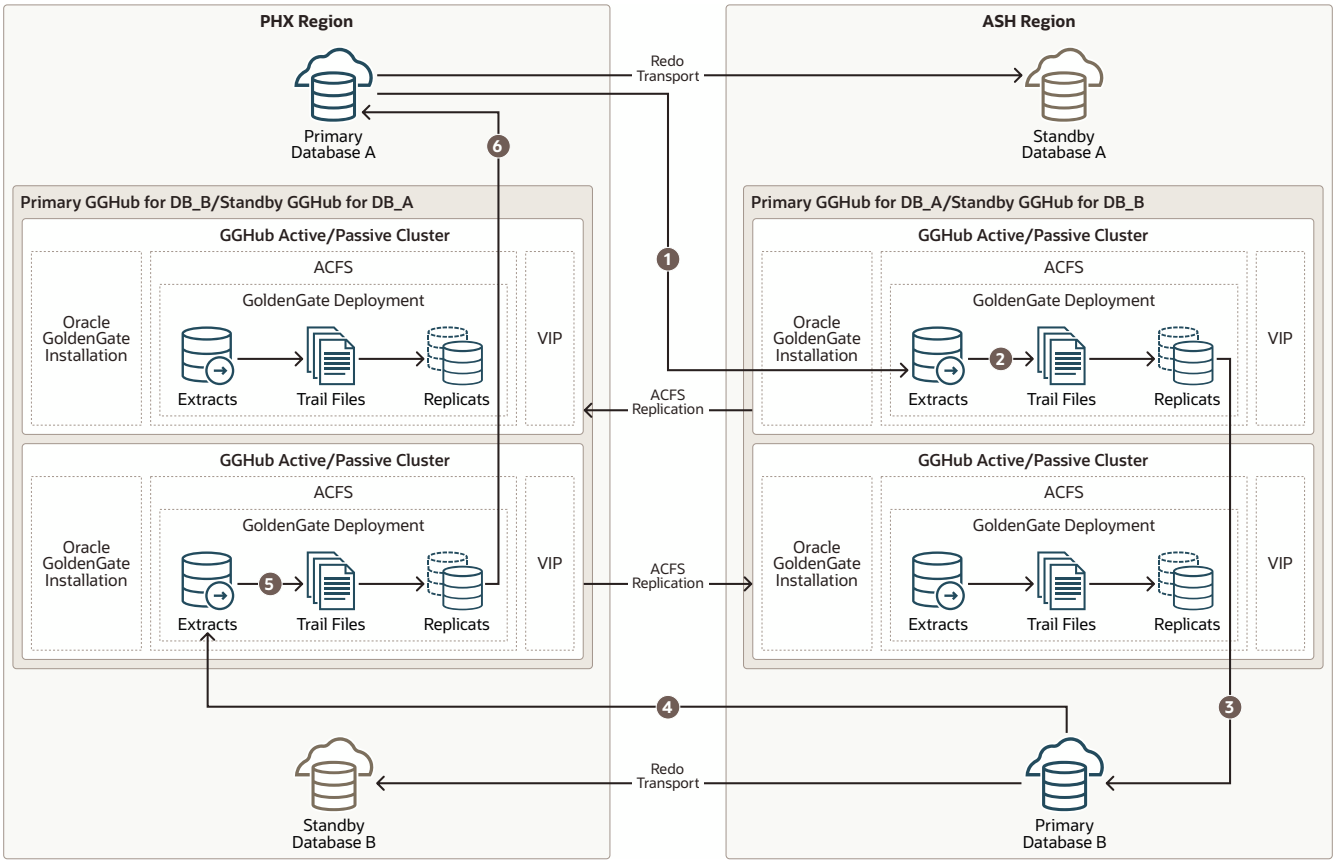
   An HA failover solution is built in to the GGHub cluster, which includes automatic failover and restart of GoldenGate processes and activity after a node failure.

   Each GGHub configuration contains a GoldenGate service manager and deployment, ACFS file system with ACFS replication, and separate application VIP.

3. Standby GGHub Active/Passive Cluster: A symmetric standby GGHub is configured. ACFS replication is set up between the primary and standby GGHubs to preserve all GoldenGate files.

   Manual GGHub failover, which includes ACFS failover, can be performed if you lose the entire primary GGHub.

Figure 20-2 Primary and Standby GGHubs in Different OCI Regions



The figure above depicts replicating data from Primary Database A to Primary Database B, and Primary B back to Primary A with the following steps:

1. Primary Database A: Primary A's Logminer server sends redo changes to an ASH region GGHub Extract process, which is on the Primary GGHub for Database A.
2. Primary GGHub: The Extract process writes changes to trail files.
3. Primary GGHub to Primary Database B: An ASH region GoldenGate Replicat process applies those changes to the target database (Primary B).
4. Primary Database B: Primary B's Logminer server sends redo to a PHX region GGHub Extract process, which is on the Primary GGHub for Database B.
5. Primary GGHub: The Extract process writes changes to trail files.
6. Primary GGHub to Primary Database A: A PHX region GoldenGate Replicat process applies those changes to the target database (Primary A).

Table 20-2 Outage Scenarios, Repair, and Restoring Redundancy for GGHubs in Different OCI Regions

Outage Scenario	Application Availability and Repair	Restoring Redundancy and Pristine State
Primary Database A (or Database B) failure	**Impact**: Near-zero application downtime. GoldenGate replication resumes when the new primary database starts. 1. One primary database is still available. All activity is routed to the existing available primary database to achieve zero application downtime. Refer to the Global Data Services Global Services Failover solution. For example, application services A-F are routed to Database A, and application services G-J are routed to Database B. If Database A fails, all services temporarily go to Database B. 2. The standby becomes the new primary automatically with Data Guard FSFO. GoldenGate replication resumes and the primary databases resynchronize. Data loss is bounded by the Data Guard protection level. If Maximum Availability or Maximum Protection is configured, zero data loss is achieved. All committed transactions are in one or both databases. Workload can be rebalanced when primary Database A and Database B are available and in sync. For example, when Database A is up and running and in sync, services A-F can go back to Database A. 3. Replicat performance will be degraded if the primary GGHub is not in the same region as the target database. Schedule a GGHub switchover with ACFS replication switchover to resume optimal Replicat performance to the target database. You may then experience two active GGHub configurations on the same GGHub cluster.	1. The old primary database is reinstated as the new standby database to restore redundancy. 2. Optionally performing a Data Guard switchover, to switch back to the original configuration, ensures that at least one primary database resides in an independent AD. Schedule a GGHub switchover with ACFS replication switchover to resume optimal Replicat performance to the target database.

Table 20-2 (Cont.) Outage Scenarios, Repair, and Restoring Redundancy for GGHubs in Different OCI Regions

Outage Scenario	Application Availability and Repair	Restoring Redundancy and Pristine State
Primary or standby GGHub single node failure	**Impact**: No application impact. GoldenGate replication resumes automatically after a couple of minutes. No action is required. An HA failover solution is built in to the GGHub that includes automatic failover and restart of GoldenGate processes and activity. Replication activity is blocked until GoldenGate processes are active again. GoldenGate Replication blackout could last a couple of minutes.	Once the node restarts, active/passive configuration is re-established.
Primary GGHub cluster crashes and is not recoverable	**Impact**: No application impact. GoldenGate replication resumes after the existing primary GGHub restarts or manual GGHub failover completes. 1. If the GGHub cluster can be restarted, then that's the simplest solution. 2. If the primary GGHub is not recoverable, then perform a manual GGHub failover to the standby GGHub, which includes ACFS failover. This typically takes several minutes. 3. Replication stops until the new primary GGhub is started, so performing step 1 or step 2 should take little time. If there's any orchestration, this should be automated.	1. If the previous GGHub eventually restarts, ACFS replication resumes in the other direction automatically. If the GGHub cluster is lost or unrecoverable, you need to rebuild a new standby GGHub. 2. Replicat performance is degraded if the primary GGHub is not in the same region as the target database. Schedule a GGHub switchover with ACFS replication switchover to resume optimal Replicat performance to the target database.
Standby GGHub cluster crashes and is not recoverable	**Impact**: No application or replication impact. 1. If the GGHub cluster can be restarted, then that's the simplest solution, and ACFS replication will resume. 2. If the standby GGHub is not recoverable, you can rebuild a new standby GGHub.	N/A

Table 20-2 (Cont.) Outage Scenarios, Repair, and Restoring Redundancy for GGHubs in Different OCI Regions

Outage Scenario	Application Availability and Repair	Restoring Redundancy and Pristine State
Complete Regional failure	**Impact**: Near Zero Application Downtime. GoldenGate replication resumes when the new primary database starts. 1. One primary database is still available. All activity is routed to the existing available primary database to achieve zero application downtime. Refer to the Global Data Services Global Services Failover solution. For example, application services A-F routed to Database A and application services G-J routed to Database B. If Database A fails, all services will temporarily go to Database B. 2. If the primary GGHub is still functional, GoldenGate replication will continue. If the primary GGHub is lost due to regional failure, then a manual GGHub failover is required. GoldenGate replication resumes and the primary databases resynchronize. Data loss is bounded by the Data Guard protection level. If Maximum availability or protection is configured, zero data loss is achieved. All committed transactions are in one or both databases. Workload can be rebalanced when Primary Database A and Database B are available and in sync. When Database A is up and running and in sync, services A-F can go back to Database A.	1. When the OCI region returns, re-establish configuration such as reinstate standby. If the previous GGHub eventually restarts, ACFS replication resumes in the other direction automatically. 2. When possible, perform a Data Guard switchover (failback) to get back to the original state where one primary database exists in each region. 3. Replicat performance is degraded if the primary GGHub is not in the same region as the target database. Schedule a GGHub switchover with ACFS replication switchover to resume optimal Replicat performance to the target database.

Task 1: Configure the Source and Target Databases for Oracle GoldenGate

The source and target Oracle GoldenGate databases should be configured using the recommendations that follow.

Step 1.1 - Configure the Databases

For Oracle Autonomous Database (ADB-D), Oracle Database 19c (19.20) or a later release is required to support parallel integrated Replicat and conflict resolution. For Oracle Exadata Database Service (ExaDB) or BaseDB, you can use any supported Oracle Database release available in the Oracle cloud.

The database configuration steps that follow are applicable for each Database Cloud Service that supports Platinum MAA solution.

The source and target Oracle GoldenGate databases should be configured using the following recommendations:

For Oracle Autonomous Database (ADB-D), you only need to add supplemental logging:

```
PDB: ALTER PLUGGABLE DATABASE ADD SUPPLEMENTAL LOG DATA;
```

For Oracle Exadata Database Service (ExaDB) or BaseDB do the following steps:

1. Enable Archivelog Mode.

   ```
   SQL> ARCHIVE LOG LIST
   Database log mode              Archive Mode
   Automatic archival             Enabled
   Archive destination            USE_DB_RECOVERY_FILE_DEST
   Oldest online log sequence     110
   Next log sequence to archive   113
   Current log sequence           113
   ```

2. Enable Force Logging.

   ```
   ALTER DATABASE FORCE LOGGING;
   ```

3. Enable GoldenGate replication.

   ```
   ALTER SYSTEM SET ENABLE_GOLDENGATE_REPLICATION=TRUE SCOPE=BOTH SID='*'
   ```

4. Add supplemental logging.

   ```
   ALTER DATABASE ADD SUPPLEMENTAL LOG DATA;
   ```

5. Configure `STREAMS_POOL_SIZE` larger to accommodate GoldenGate.

   Use this formula to calculate the appropriate `STREAMS_POOL_SIZE` value:

   ```
   STREAMS_POOL_SIZE = (((#Extracts + #Integrated Replicats) * 1GB) * 1.25)
   ```

   For example, in a database with 2 Extracts and 2 integrated Replicats:

   ```
   STREAMS_POOL_SIZE = 4GB * 1.25 = 5GB
   ```

   And the parameter is set:

   ```
   ALTER SYSTEM SET STREAMS_POOL_SIZE=5G SCOPE=BOTH SID='*';
   ```

For the steps to prepare the database for Oracle GoldenGate, see Preparing the Database for Oracle GoldenGate.

Step 1.2 - Create the Database Replication Administrator User

For ADB-D

For ADB-D deployments, you only need to issue

```
ALTER USER ggadmin IDENTIFIED BY <password> ACCOUNT UNLOCK;
```

For Oracle Exadata Database Service (ExaDB) or BaseDB

The following steps are only applicable for Oracle Exadata Database Service (ExaDB) or BaseDB.

The source and target databases need a GoldenGate administrator user created, with appropriate privileges assigned as follows:

- For the multitenant container database (CDB):
  - Source database, GoldenGate Extract must be configured to connect to a user in the root container database, using a c##
  - Target database, a separate GoldenGate administrator user is needed for each pluggable database (PDB).
  - For details about creating a GoldenGate administrator in an Oracle Multitenant Database, see Configuring Oracle GoldenGate in a Multitenant Container Database.
- For non-CDB databases, see Establishing Oracle GoldenGate Credentials

1. As the `oracle` OS user on the source database system, execute the following SQL instructions to create the database user for Oracle GoldenGate and assign the required privileges:

```
[opc@exadb1_node1 ~]$ sudo su - oracle
[oracle@exadb1_node1 ~]$ source dbName.env
[oracle@exadb1_node1 ~]$ sqlplus / as sysdba

# Source CDB
SQL>
alter session set container=cdb$root;
create user c##ggadmin identified by "ggadmin_password" container=all
default tablespace USERS temporary tablespace temp;
alter user c##ggadmin quota unlimited on users;
grant set container to c##ggadmin container=all;
grant alter system to c##ggadmin container=all;
grant create session to c##ggadmin container=all;
grant alter any table to c##ggadmin container=all;
grant resource to c##ggadmin container=all;
exec
dbms_goldengate_auth.grant_admin_privilege('c##ggadmin',container=>'all');

# Source PDB
SQL>
alter session set container=pdbName;
create user ggadmin identified by "ggadmin_password" container=current;
```

```
grant create session to ggadmin container=current;
grant alter any table to ggadmin container=current;
grant resource to ggadmin container=current;
exec dbms_goldengate_auth.grant_admin_privilege('ggadmin');
```

2. As the `oracle` OS user on the target system, execute the following SQL instructions to create the database user for Oracle GoldenGate and assign the required privileges:

```
[opc@exadb2_node1 ~]$ sudo su - oracle
[oracle@exadb2_node1 ~]$ source dbName.env
[oracle@exadb2_node1 ~]$ sqlplus / as sysdba

# Target PDB
SQL>
alter session set container=pdbName;
create user ggadmin identified by "ggadmin_password" container=current;
grant alter system to ggadmin container=current;
grant create session to ggadmin container=current;
grant alter any table to ggadmin container=current;
grant resource to ggadmin container=current;
grant dv_goldengate_admin, dv_goldengate_redo_access to ggadmin
container=current;
exec dbms_goldengate_auth.grant_admin_privilege('ggadmin');
```

Step 1.3 - Create the Database Services

> **Note:**
>
> This step is not required for ADB-D deployments.

If the source and target databases are running the recommended configuration on an Oracle RAC cluster with Oracle Data Guard, a role-based service must be created that allows the Extract or Replicat processes to connect to the correct Data Guard primary database instance.

When using a source multitenant database, a separate service is required for the root container database (CDB) and the pluggable database (PDB) that contains the schema being replicated. For a target multitenant database, a single service is required for the PDB.

1. As the `oracle` OS user on the primary database system, use dbaascli to find the CDB and PDB name, as shown here:

```
[opc@exadb1_node1 ~]$ sudo su - oracle
[oracle@exadb1_node1 ~]$ source dbName.env
[oracle@exadb1_node1 ~]$ dbaascli database getDetails
 --dbname dbName |egrep 'dbName|pdbName'

  "dbName" : "dbName",
      "pdbName" : "pdbName",
```

2. As the `oracle` OS user on the primary and standby database systems, create and start the CDB database service using the following command:

```
[opc@exadb1_node1 ~]$ sudo su - oracle
[oracle@exadb1_node1 ~]$ source dbName.env
[oracle@exadb1_node1 ~]$ srvctl add service -db $ORACLE_UNQNAME
 -service dbName.goldengate.com -preferred ORACLE_SID1
 -available ORACLE_SID2 -role PRIMARY
```

3. As the `oracle` OS user on the primary and standby database systems, create and start the PDB database service using the following command:

```
[oracle@exadb1_node1 ~]$ srvctl add service -db $ORACLE_UNQNAME
 -service dbName.pdbName.goldengate.com -preferred ORACLE_SID1
 -available ORACLE_SID2 -pdb pdbName -role PRIMARY
```

4. As the `oracle` OS user on the primary and standby database systems, start and verify that the services are running, as shown here:

```
[oracle@exadb1_node1 ~]$ srvctl start service -db $ORACLE_UNQNAME -role
[oracle@exadb1_node1 ~]$ srvctl status service -d $ORACLE_UNQNAME |grep
goldengate

Service dbName.goldengate.com  is running on instance(s) SID1
Service dbName.pdbName.goldengate.com is running on instance(s) SID1
```

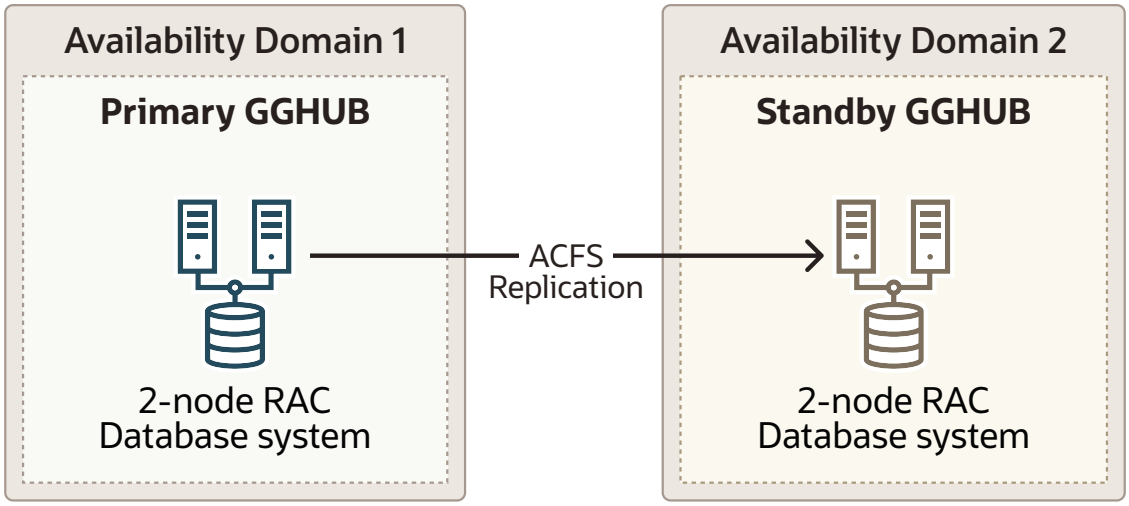
> **Note:**
>
> Repeat all of Step 1.3 in the source and target database system.

Task 2: Prepare a Primary and Standby Base System for GGHub

Step 2.1 - Deploy an Oracle RAC 2-Node Cluster System

Deploy a minimum of two GGHubs (primary and standby). Each GGHub must be deployed as a 2-node Oracle RAC database system as described in Oracle Base Database Service.

Figure 20-3 Oracle GoldenGate Hub Hardware Architecture



Step 2.2 - Remove the Standard Database and Rearrange the Disk Group Layout

1. As the `oracle` OS user on the first GGHub node, remove the standard database:

```
[opc@gghub_prim1 ~]$ sudo su - oracle
[oracle@gghubN-node1 ~]$ dbca -deleteDatabase -silent -
sourceDB $ORACLE_UNQNAME
Enter SYS user password: ###############

[WARNING] [DBT-19202] The Database Configuration Assistant will delete the
Oracle instances and datafiles for your database. All information in the
database will be destroyed.
Prepare for db operation
32% complete
Connecting to database
39% complete
...
100% complete
Database deletion completed.
Look at the log file "/u01/app/oracle/cfgtoollogs/dbca/DB0502_fra2pr/
DB0502_fra2pr.log" for further details.
```

2. As the `grid` OS user on the second GGHub node, dismount the RECO diskgroup:

```
[opc@gghub_prim2 ~]$ sudo su - grid
[grid@gghub_prim2 ~]$ sqlplus / as sysasm

SQL> alter diskgroup RECO dismount;
```

3. As the `grid` OS user on the first **gghub** node, drop the RECO diskgroup and assign the disks to the DATA diskgroup:

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ sqlplus / as sysasm

SQL>
drop diskgroup RECO INCLUDING CONTENTS;
alter diskgroup DATA add disk '/dev/RECODISK1';
alter diskgroup DATA add disk '/dev/RECODISK2';
alter diskgroup DATA add disk '/dev/RECODISK3';
alter diskgroup DATA add disk '/dev/RECODISK4';
```

4. As the `root` OS user on all GGhub nodes, reboot the node:

```
[opc@gghub_prim1 ~]$ sudo reboot
```

> **Note:**
>
> Repeat this step in the primary and standby GGHubs.

Step 2.3 - Download the Required Software

1. As the `opc` OS user on all GGHub nodes, create the staging and scripts directories:

```
[opc@gghub_prim1 ~]$
sudo mkdir -p /u01/oracle/stage
sudo mkdir /u01/oracle/scripts
sudo chown -R oracle:oinstall /u01/oracle
sudo chmod -R g+w /u01/oracle
sudo chmod -R o+w /u01/oracle/stage
```

2. As the `opc` OS user on all GGHub nodes, download the following software in the directory /u01/oracle/stage:
   - Download the latest Oracle GoldenGate 21c (or later release) Microservices software from Oracle GoldenGate Downloads.
   - Download subsequent patches to the base release from the **Patches and Updates** tab of My Oracle Support.
     - See Installing Patches for Oracle GoldenGate Microservices Architecture for more information.
     - Minimum required version is Patch 35214851: Oracle GoldenGate 21.9.0.0.2 Microservices for Oracle
   - Download the latest OPatch release, Patch 6880880, for Oracle Database 21c (21.0.0.0.0) from My Oracle Support Document 2542082.1.
   - Download the Oracle Grid Infrastructure Standalone Agents for Oracle Clusterware 19c, release 10.2 or later, from Oracle Grid Infrastructure Standalone Agents for Oracle Clusterware.

- Download the python script (`secureServices.py`) from My Oracle Support Document 2826001.1
- Download the Oracle GGHUB Scripts from My Oracle Support Document 2951572.1

3. As the `grid` OS user on all GGHub nodes, unzip the GGhub scripts file downloaded from My Oracle Support Document 2951572.1 into the directory `/u01/oracle/scripts`.

   Place the script in the same location on all primary and standby GGhub nodes

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ unzip -q /u01/oracle/stage/
gghub_scripts_YYYYYMMDD.zip -d /u01/oracle/scripts/
```

Step 2.4 - Configure Oracle Linux To Use the Oracle Public YUM Repository

The Oracle Linux yum server hosts software for Oracle Linux and compatible distributions. These instructions help you get started configuring your Linux system for Oracle Linux yum server and installing software through yum.

- As the `root` OS user in all GGHub systems, create the file `/etc/yum.repos.d/oracle-public-yum-ol7.repo` with the following contents:

```
[opc@gghub_prim1 ~]$ sudo su -
[root@gghub_prim1 ~]#
cat > /etc/yum.repos.d/oracle-public-yum-ol7.repo <<EOF
[ol7_latest]
name=Oracle Linux $releasever Latest ($basearch)
baseurl=http://yum$ociregion.oracle.com/repo/OracleLinux/OL7/latest/
\$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
gpgcheck=1
enabled=1
EOF
```

Task 3: Configure Oracle GoldenGate for the Primary and Standby GGHub

Step 3.1 - Install Oracle GoldenGate Software

Install Oracle GoldenGate software locally on all nodes of the primary and standby GGHub configuration that will be part of the GoldenGate configuration. Make sure the installation directory is identical on all nodes.

Perform the following sub-steps to complete this step:

- Step 3.1.1 Unzip the Software and Create the Response File for the Installation
- Step 3.1.2 Install Oracle GoldenGate Software
- Step 3.1.3 Installing Patches for Oracle GoldenGate Microservices Architecture

Step 3.1.1 Unzip the Software and Create the Response File for the Installation

As the `oracle` OS user on all GGHub nodes, unzip the Oracle GoldenGate software:

```
[opc@gghub_prim1 ~]$ sudo su - oracle
[oracle@gghub_prim1 ~]$ unzip -q
 /u01/oracle/stage/p36175132_21130000GGRU_Linux-x86-64.zip -d
 /u01/oracle/stage
```

The software includes an example response file for Oracle Database 21c and earlier supported versions. Copy the response file to a shared file system, so the same file can be used to install Oracle GoldenGate on all database nodes, and edit the following parameters:

- `INSTALL_OPTION=ora21c`
- `SOFTWARE_LOCATION=/u01/app/oracle/goldengate/gg21c (recommended location)`

As the `oracle` OS user on all GGHub nodes, copy and edit the response file for the installation:

```
[oracle@gghub_prim1 ~]$ cp
 /u01/oracle/stage/fbo_ggs_Linux_x64_Oracle_services_shiphome/Disk1/response/
oggcore.rsp
 /u01/oracle/stage
[oracle@gghub_prim1 ~]$ vi /u01/oracle/stage/oggcore.rsp

# Before
INSTALL_OPTION=
SOFTWARE_LOCATION=

# After
INSTALL_OPTION=ora21c
SOFTWARE_LOCATION=/u01/app/oracle/goldengate/gg21c
```

Step 3.1.2 Install Oracle GoldenGate Software

As the `oracle` OS user on all GGHub nodes, run `runInstaller` to install Oracle GoldenGate:

```
[oracle@gghub_prim1 ~]$ cd
 /u01/oracle/stage/fbo_ggs_Linux_x64_Oracle_services_shiphome/Disk1/
[oracle@gghub_prim1 ~]$ ./runInstaller -silent -nowait
 -responseFile /u01/oracle/stage/oggcore.rsp

Starting Oracle Universal Installer...

Checking Temp space: must be greater than 120 MB.   Actual 32755 MB    Passed
Checking swap space: must be greater than 150 MB.   Actual 16383 MB    Passed
Preparing to launch Oracle Universal Installer from
 /tmp/OraInstall2022-07-08_02-54-51PM.
 Please wait ...
You can find the log of this install session at:
 /u01/app/oraInventory/logs/installActions2022-07-08_02-54-51PM.log
Successfully Setup Software.
The installation of Oracle GoldenGate Services was successful.
Please check
 '/u01/app/oraInventory/logs/silentInstall2022-07-08_02-54-51PM.log'
 for more details.

[oracle@gghub_prim1 ~]$ cat
```

```
 /u01/app/oraInventory/logs/silentInstall2022-07-08_02-54-51PM.log

The installation of Oracle GoldenGate Services was successful.
```

Step 3.1.3 Installing Patches for Oracle GoldenGate Microservices Architecture

As the `oracle` OS user on all GGHub nodes, install the latest OPatch:

```
[oracle@gghub_prim1 ~]$ unzip -oq -d
 /u01/app/oracle/goldengate/gg21c
 /u01/oracle/stage/p6880880_210000_Linux-x86-64.zip
[oracle@gghub_prim1 ~]$ cat >> ~/.bashrc <<EOF
export ORACLE_HOME=/u01/app/oracle/goldengate/gg21c
export PATH=$ORACLE_HOME/OPatch:$PATH
EOF
[oracle@gghub_prim1 ~]$ . ~/.bashrc
[oracle@gghub_prim1 ~]$ opatch lsinventory |grep
 'Oracle GoldenGate Services'

Oracle GoldenGate Services                                        21.1.0.0.0


[oracle@gghub_prim1 Disk1]$ opatch version
OPatch Version: 12.2.0.1.37

OPatch succeeded.
```

As the `oracle` OS user on all GGHub nodes, run OPatch `prereq` to validate any conflict before applying the patch:

```
[oracle@gghub_prim1 ~]$ unzip -oq -d /u01/oracle/stage/
 /u01/oracle/stage/p35214851_2190000GGRU_Linux-x86-64.zip
[oracle@gghub_prim1 ~]$ cd /u01/oracle/stage/35214851/
[oracle@gghub_prim1 35214851]$ opatch prereq
 CheckConflictAgainstOHWithDetail -ph ./

Oracle Interim Patch Installer version 12.2.0.1.26
Copyright (c) 2023, Oracle Corporation.  All rights reserved.

PREREQ session

Oracle Home       : /u01/app/oracle/goldengate/gg21c
Central Inventory : /u01/app/oraInventory
   from           : /u01/app/oracle/goldengate/gg21c/oraInst.loc
OPatch version    : 12.2.0.1.26
OUI version       : 12.2.0.9.0
Log file location :
 /u01/app/oracle/goldengate/gg21c/cfgtoollogs/opatch/
opatch2023-04-21_13-44-16PM_1.log

Invoking prereq "checkconflictagainstohwithdetail"

Prereq "checkConflictAgainstOHWithDetail" passed.
```

```
OPatch succeeded.
```

As the `oracle` OS user on all GGHub nodes, patch Oracle GoldenGate Microservices Architecture using OPatch:

```
[oracle@gghub_prim1 35214851]$ opatch apply

Oracle Interim Patch Installer version 12.2.0.1.37
Copyright (c) 2023, Oracle Corporation.  All rights reserved.

Oracle Home       : /u01/app/oracle/goldengate/gg21c
Central Inventory : /u01/app/oraInventory
   from           : /u01/app/oracle/goldengate/gg21c/oraInst.loc
OPatch version    : 12.2.0.1.37
OUI version       : 12.2.0.9.0
Log file location :
 /u01/app/oracle/goldengate/gg21c/cfgtoollogs/opatch/
opatch2023-04-21_19-40-41PM_1.log
Verifying environment and performing prerequisite checks...
OPatch continues with these patches:   35214851

Do you want to proceed? [y|n]
y
User Responded with: Y
All checks passed.

Please shutdown Oracle instances running out of this ORACLE_HOME on
 the local system.
(Oracle Home = '/u01/app/oracle/goldengate/gg21c'

Is the local system ready for patching? [y|n]
y
User Responded with: Y
Backing up files...
Applying interim patch '35214851' to OH '/u01/app/oracle/goldengate/gg21c'

Patching component oracle.oggcore.services.ora21c, 21.1.0.0.0...
Patch 35214851 successfully applied.
Log file location:
 /u01/app/oracle/goldengate/gg21c/cfgtoollogs/opatch/
opatch2023-04-21_19-40-41PM_1.log

OPatch succeeded.

[oracle@gghub_prim1 35214851]$ opatch lspatches
35214851;

OPatch succeeded.
```

> **Note:**
>
> Repeat all of the steps in step 3.1 for the primary and standby GGHub systems.

Step 3.2 - Configure the Cloud Network

You must configure virtual cloud network (VCN) components such as private DNS zones, VIP, bastion, security lists, and firewalls for Oracle GoldenGate to function correctly.

To learn more about VCNs and security lists, including instructions for creating them, see the Oracle Cloud Infrastructure Networking documentation.

Perform the following sub-steps to complete this step:

- Step 3.2.1 - Create an Application Virtual IP Address (VIP) for GGhub
- Step 3.2.2 - Add an Ingress Rule for port 443
- Step 3.2.3 - Open Port 443 in the GGhub Firewall
- Step 3.2.4 - Configure Network Connectivity Between the Primary and Standby GGHUB Systems
- Step 3.2.5 - Configure Private DNS Zones Views and Resolvers

Step 3.2.1 - Create an Application Virtual IP Address (VIP) for GGhub

A dedicated application VIP is required to allow access to the GoldenGate Microservices using the same host name, regardless of which node of the cluster is hosting the services. The VIP is assigned to the GGHUB system and is automatically migrated to another node in the event of a node failure. Two VIPs are required, one for the primary and another one for the standby GGHUBs.

As the `grid` OS user on all GGhub nodes, run the following commands to get the `vnicId` of the Private Endpoint in the same subnet at resource `ora.net1.network`:

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ crsctl status resource -p -attr NAME,USR_ORA_SUBNET
 -w "TYPE = ora.network.type" |sort | uniq

NAME=ora.net1.network
USR_ORA_SUBNET=10.60.2.0

[grid@gghub_prim1 ~]$ curl 169.254.169.254/opc/v1/vnics

[
  {
    "macAddr": "02:00:17:04:70:AF",
    "privateIp": "10.60.2.120",
    "subnetCidrBlock": "10.60.2.0/24",
    "virtualRouterIp": "10.60.2.1",
    "vlanTag": 3085,
    "vnicId": "ocid1.vnic.oc1.eu-frankfurt-1.ocid_value"
  },
  {
    "macAddr": "02:00:17:08:69:6E",
    "privateIp": "192.168.16.18",
    "subnetCidrBlock": "192.168.16.16/28",
    "virtualRouterIp": "192.168.16.17",
    "vlanTag": 879,
    "vnicId": "ocid1.vnic.oc1.eu-frankfurt-1.ocid_value"
  }
```

```
[grid@gghub_prim2 ~]$ curl 169.254.169.254/opc/v1/vnics

[
  {
    "macAddr": "00:00:17:00:C9:19",
    "privateIp": "10.60.2.148",
    "subnetCidrBlock": "10.60.2.0/24",
    "virtualRouterIp": "10.60.2.1",
    "vlanTag": 572,
    "vnicId": "ocid1.vnic.oc1.eu-frankfurt-1.ocid_value"
  },
  {
    "macAddr": "02:00:17:00:84:B5",
    "privateIp": "192.168.16.19",
    "subnetCidrBlock": "192.168.16.16/28",
    "virtualRouterIp": "192.168.16.17",
    "vlanTag": 3352,
    "vnicId": "ocid1.vnic.oc1.eu-frankfurt-1.ocid_value"
  }
```

> **Note:**
>
> For the next step, you will need to use the Cloud Shell to assign the private IP to the GGHUB nodes. See Using Cloud Shell for more information.

As your user on the cloud shell, run the following commands to assign the private IP to the GGHUB nodes:

```
username@cloudshell:~ (eu-frankfurt-1)$ export node1_vnic=
'ocid1.vnic.oc1.eu-
frankfurt-1.abtheljrl5udtgryrscypy5btmlfncawqkjlcql3kkpj64e2lb5xbmbrehkq'
username@cloudshell:~ (eu-frankfurt-1)$ export node2_vnic=
'ocid1.vnic.oc1.eu-
frankfurt-1.abtheljre6rf3xoxtgl2gam3lav4vcyftz5fppm2ciin4wzjxucalzj7b2bq'
username@cloudshell:~ (eu-frankfurt-1)$ export ip_address='10.60.2.65'
username@cloudshell:~ (eu-frankfurt-1)$ oci network vnic assign-private-ip
 --unassign-if-already-assigned --vnic-id $node1_vnic --ip-address $ip_address
username@cloudshell:~ (eu-frankfurt-1)$ oci network vnic assign-private-ip
 --unassign-if-already-assigned --vnic-id $node2_vnic --ip-address $ip_address

Example of the output:
{
  "data": {
    "availability-domain": null,
    "compartment-id": "ocid1.compartment.oc1..ocid_value",
    "defined-tags": {},
    "display-name": "privateip20230292502117",
    "freeform-tags": {},
    "hostname-label": null,
    "id": "ocid1.privateip.oc1.eu-frankfurt-1.ocid_value",
    "ip-address": "10.60.2.65",
    "is-primary": false,
    "subnet-id": "ocid1.subnet.oc1.eu-frankfurt-1.ocid_value",
```

```
    "time-created": "2023-07-27T10:21:17.851000+00:00",
    "vlan-id": null,
    "vnic-id": "ocid1.vnic.oc1.eu-frankfurt-1.ocid_value"
  },
  "etag": "da972988"
}
```

As the `root` OS user on the first GGhub node, run the following command to create the application VIP managed by Oracle Clusterware:

```
[opc@gghub_prim1 ~]$ sudo su -
[root@gghub_prim1 ~]# sh /u01/oracle/scripts/add_appvip.sh

Application VIP Name: gghub_prim_vip
Application VIP Address: 10.60.2.65
Using configuration parameter file:
 /u01/app/19.0.0.0/grid/crs/install/crsconfig_params
The log of current session can be found at:
  /u01/app/grid/crsdata/gghublb1/scripts/appvipcfg.log
```

> **Note:**
>
> Repeat all the steps in step 3.2.1 for the primary and standby GGHUB systems.

Step 3.2.2 - Add the Ingress Security List Rules

Using the Cloud Console, add two ingress security list rules in the Virtual Cloud Network (VCN) assigned to the GGhub.

One ingress rule is for TCP traffic on destination port 443 from authorized source IP addresses and any source port to connect to the Oracle GoldenGate service using NGINX as a reverse proxy, and the other is for allowing ICMP TYPE 8 (ECHO) between the primary and standby GGhubs required to enable ACFS replication. For more information, see Working with Security Lists and My Oracle Support Document 2584309.1.

After you update the security list, it will have an entry with values similar to the following ones:

1. NGINX - TCP 443
   - Source Type: CIDR
   - Source CIDR: 0.0.0.0/0
   - IP Protocol: TCP
   - Source Port Range: All
   - Destination Port Range: 443
   - Allows: TCP traffic for ports: 443 HTTPS
   - Description: Oracle GoldenGate 443
2. ACFS - ICMP TYPE 8 (ECHO)
   - Source Type: CIDR
   - Source CIDR: 0.0.0.0/0
   - IP Protocol: ICMP

- Allows: ICMP traffic for: 8 Echo
- Description: Required for ACFS replication

Step 3.2.3 - Open Port 443 in the GGhub Firewall

As the `opc` OS user on all GGhub nodes of the primary and standby system, add the required rules to IPTables:

```
[opc@gghub_prim1 ~]$ sudo vi /etc/sysconfig/iptables

-A INPUT -p tcp -m state --state NEW -m tcp --dport 443 -j ACCEPT
 -m comment --comment "Required for access to GoldenGate, Do not remove
 or modify. "
-A INPUT -p tcp -m state --state NEW -m tcp --match multiport
 --dports 9100:9105 -j ACCEPT -m comment --comment "Required for access
 to GoldenGate, Do not remove or modify. "

[opc@gghub_prim1 ~]$ sudo systemctl restart iptables
```

> **Note:**
>
> See Implementing Oracle Linux Security for more information.

Step 3.2.4 - Configure Network Connectivity Between the Primary and Standby GGHUB Systems

Oracle ACFS snapshot-based replication uses ssh as the transport between the primary and standby clusters. To support ACFS replication, `ssh` must be usable in either direction between the clusters — from the primary cluster to the standby cluster and from the standby to the primary. See Configuring ssh for Use With Oracle ACFS Replication in *Oracle Automatic Storage Management Administrator's Guide*.

To learn more about whether subnets are public or private, including instructions for creating the connection, see section Connectivity Choices in the Oracle Cloud Infrastructure Networking documentation.

Step 3.2.5 - Configure Private DNS Zones Views and Resolvers

You must create a private DNS zone view and records for each application VIP. This is required for the primary GGHUB to reach the standby GGHUB deployment VIP host name.

Follow the steps in Configure private DNS zones views and resolvers to create your private DNS zone and a record entry for each dedicated GGHUB application virtual IP address (VIP) created in Step 3.2.1.

As the `opc` OS user on any GGhub node, validate that all application VIPs can be resolved:

```
[opc@gghub_prim1 ~]$ nslookup
 gghub_prim_vip.frankfurt.goldengate.com |tail -2

Address: 10.60.2.120

[opc@gghub_prim1 ~]$ nslookup
 gghub_stby_vip.frankfurt.goldengate.com |tail -2
```

```
Address: 10.60.0.185
```

Step 3.3 - Configure ACFS File System Replication Between GGHubs in the Same Region

Oracle GoldenGate Microservices Architecture is designed with a simplified installation and deployment directory structure. The installation directory: should be placed on local storage on each database node to minimize downtime during software patching. The deployment directory: which is created during deployment creation using the Oracle GoldenGate Configuration Assistant (oggca.sh), must be placed on a shared file system. The deployment directory contains configuration, security, log, parameter, trail, and checkpoint files. Placing the deployment in Oracle Automatic Storage Management Cluster File system (ACFS) provides the best recoverability and failover capabilities in the event of a system failure. Ensuring the availability of the checkpoint files cluster-wide is essential so that the GoldenGate processes can continue running from their last known position after a failure occurs.

It is recommended that you allocate enough trail file disk space for a minimum of 12 hours of trail files. Doing this will give sufficient space for trail file generation should a problem occur with the target environment that prevents it from receiving new trail files. The amount of space needed for 12 hours can only be determined by testing trail file generation rates with real production data. If you want to build contingency for a long planned maintenance event of one of the GoldenGate Primary Database or systems, you can allocate sufficient ACFS space for 2 days. Monitoring space utilization is always recommended regardless of how much space is allocated.

> **Note:**
>
> If the GoldenGate hub will support multiple service manager deployments using separate ACFS file systems, the following steps should be repeated for each file ACFS file system.

Perform the following sub-steps to complete this step:

- Step 3.3.1 - Create the ASM File system
- Step 3.3.2 - Create the Cluster Ready Services (CRS) Resource
- Step 3.3.3 - Verify the Currently Configured ACFS File System
- Step 3.3.4 - Start and Check the Status of the ACFS Resource
- Step 3.3.5 – Create CRS Dependencies Between ACFS and an Application VIP
- Step 3.3.6 – Create the SSH Daemon CRS Resource
- Step 3.3.7 – Enable ACFS Replication
- Step 3.3.8 – Create the ACFS Replication CRS Action Scripts

Step 3.3.1 - Create the ASM File system

As the `grid` OS user on the first GGHUB node, use `asmcmd` to create the ACFS volume:

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ asmcmd volcreate -G DATA -s 120G ACFS_GG1
```

> **Note:**
>
> Modify the file system size according to the determined size requirements.

As the `grid` OS user on the first GGHUB node, use `asmcmd` to confirm the "Volume Device":

```
[grid@gghub_prim1 ~]$ asmcmd volinfo -G DATA ACFS_GG1

Diskgroup Name: DATA
         Volume Name: ACFS_GG1
         Volume Device: /dev/asm/acfs_gg1-256
         State: ENABLED
         Size (MB): 1228800
         Resize Unit (MB): 64
         Redundancy: UNPROT
         Stripe Columns: 8
         Stripe Width (K): 1024
         Usage:
         Mountpath:
```

As the `grid` OS user on the first GGHUB node, format the partition with the following `mkfs` command:

```
[grid@gghub_prim1 ~]$ /sbin/mkfs -t acfs /dev/asm/acfs_gg1-256

mkfs.acfs: version                   = 19.0.0.0.0
mkfs.acfs: on-disk version           = 46.0
mkfs.acfs: volume                    = /dev/asm/acfs_gg1-256
mkfs.acfs: volume size               = 128849018880  ( 120.00 GB )
mkfs.acfs: Format complete.
```

Step 3.3.2 - Create the Cluster Ready Services (CRS) Resource

As the `opc` OS user on all GGHUB nodes, create the ACFS mount point:

```
[opc@gghub_prim1 ~]$ sudo mkdir -p /mnt/acfs_gg1
[opc@gghub_prim1 ~]$ sudo chown oracle:oinstall /mnt/acfs_gg1
```

Create the file system resource as the root user. Due to the implementation of distributed file locking on ACFS, unlike DBFS, it is acceptable to mount ACFS on more than one GGhub node at any one time.

As the `root` OS user on the first GGHUB node, create the CRS resource for the new ACFS file system:

```
[opc@gghub_prim1 ~]$ sudo su -
[root@gghub_prim1 ~]#

cat > /u01/oracle/scripts/add_asm_filesystem.sh <<EOF
# Run as ROOT
$(grep ^crs_home /etc/oracle/olr.loc | cut -d= -f2)/bin/srvctl
 add filesystem \
```

```
-device /dev/asm/<acfs_volume> \
-volume ACFS_GG1 \
-diskgroup DATA \
-path /mnt/acfs_gg1 -user oracle \
-node gghub_prim1,gghub_prim2 \
-autostart NEVER \
-mountowner oracle \
-mountgroup oinstall \
-mountperm 755
EOF
[root@gghub_prim1 ~]# sh /u01/oracle/scripts/add_asm_filesystem.sh
```

Step 3.3.3 - Verify the Currently Configured ACFS File System

As the `grid` OS user on the first GGHUB node, use the following command to validate the file system details:

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ srvctl config filesystem -volume ACFS_GG1
 -diskgroup DATA


Volume device: /dev/asm/acfs_gg1-256
Diskgroup name: data
Volume name: acfs_gg1
Canonical volume device: /dev/asm/acfs_gg1-256
Accelerator volume devices:
Mountpoint path: /mnt/acfs_gg1
Mount point owner: oracle
Mount point group: oinstall
Mount permissions: owner:oracle:rwx,pgrp:oinstall:r-x,other::r-x
Mount users: grid
Type: ACFS
Mount options:
Description:
Nodes: gghub_prim1 gghub_prim2
Server pools: *
Application ID:
ACFS file system is enabled
ACFS file system is individually enabled on nodes:
ACFS file system is individually disabled on nodes:
```

Step 3.3.4 - Start and Check the Status of the ACFS Resource

As the `grid` OS user on the first gghub node, use the following command to start and check the file system:

```
[grid@gghub_prim1 ~]$ srvctl start  filesystem -volume ACFS_GG1
 -diskgroup DATA -node `hostname`
[grid@gghub_prim1 ~]$ srvctl status filesystem -volume ACFS_GG1
 -diskgroup DATA
ACFS file system /mnt/acfs_gg1 is mounted on nodes gghub_prim1
```

The CRS resource created is named using the format `ora.diskgroup_name.volume_name.acfs`. Using the above file system example, the CRS resource is called `ora.data.acfs_gg.acfs`.

As the `grid` OS user on the first gghub node, use the following command to see the ACFS resource in CRS:

```
[grid@gghub_prim1 ~]$ crsctl stat res ora.data.acfs_gg1.acfs

NAME=ora.data.acfs_gg1.acfs
TYPE=ora.acfs_cluster.type
TARGET=ONLINE
STATE=ONLINE on gghub_prim1
```

Step 3.3.5 – Create CRS Dependencies Between ACFS and an Application VIP

To ensure that the file system is mounted on the same Oracle GGHub node as the VIP, add the VIP CRS resource as a dependency to the ACFS resource, using the following example commands. Each separate replicated ACFS file system will have its own dedicated VIP.

1. As the `root` OS user on the first GGHub node, use the following command to determine the current start and stop dependencies of the VIP resource:

```
[opc@gghub_prim1 ~]$ sudo su -
[root@gghub_prim1 ~]# $(grep ^crs_home /etc/oracle/olr.loc | cut -d=
 -f2)/bin/crsctl stat res -w "TYPE co appvip" |grep NAME | cut -f2 -d"="
gghub_prim_vip1

[root@gghub_prim1 ~]# export APPVIP=gghub_prim_vip1
[root@gghub_prim1 ~]# $(grep ^crs_home /etc/oracle/olr.loc | cut -d=
 -f2)/bin/crsctl stat res $APPVIP -f|grep _DEPENDENCIES

START_DEPENDENCIES=hard(ora.net1.network) pullup(ora.net1.network)
STOP_DEPENDENCIES=hard(intermediate:ora.net1.network)
```

2. As the `root` OS user on the first GGHub node, determine the ACFS file system name:

```
[root@gghub_prim1 ~]# $(grep ^crs_home /etc/oracle/olr.loc | cut -d=
 -f2)/bin/crsctl stat res -w "NAME co acfs_gg1" |grep NAME

NAME=ora.data.acfs_gg.acfs

[root@gghub_prim1 ~]# export ACFS_NAME='ora.data.acfs_gg1.acfs'
```

3. As the `root` OS user on the first GGHub node, modify the start and stop dependencies of the VIP resource:

```
[root@gghub_prim1 ~]# $(grep ^crs_home /etc/oracle/olr.loc | cut -d=
 -f2)/bin/crsctl modify res $APPVIP -attr
 "START_DEPENDENCIES='hard(ora.net1.network,$ACFS_NAME)
pullup(ora.net1.network)

pullup:always($ACFS_NAME)',STOP_DEPENDENCIES='hard(intermediate:ora.net1.ne
twork,$ACFS_NAME)',HOSTING_MEMBERS=,PLACEMENT=balanced"
```

4. As the `grid` OS user on the first GGHub node, start the VIP resource:

```
[grid@gghub_prim1 ~]$ $(grep ^crs_home /etc/oracle/olr.loc | cut -d=
 -f2)/bin/crsctl stat res -w "TYPE co appvip" |grep NAME | cut -f2 -d"="
gghub_prim_vip1

[grid@gghub_prim1 ~]$ export APPVIP=gghub_prim_vip1

[grid@gghub_prim1 ~]$ crsctl start resource $APPVIP
CRS-2672: Attempting to start 'gghub_prim_vip1' on 'gghub_prim1'
CRS-2676: Start of 'gghub_prim_vip1' on 'gghub_prim1' succeeded
```

> **Note:**
>
> Before moving to the next step, it is important to ensure the VIP can be mounted on both GGHub nodes.

5. As the `grid` OS user on the first GGHub node, relocate the VIP resource:

```
[grid@gghub_prim1 ~]$ crsctl relocate resource $APPVIP -f


CRS-2673: Attempting to stop 'gghub_prim_vip1' on 'gghub_prim1'
CRS-2677: Stop of 'gghub_prim_vip1' on 'gghub_prim1' succeeded
CRS-2673: Attempting to stop 'ora.data.acfs_gg1.acfs' on 'gghub_prim1'
CRS-2677: Stop of 'ora.data.acfs_gg1.acfs' on 'gghub_prim1' succeeded
CRS-2672: Attempting to start 'ora.data.acfs_gg1.acfs' on 'gghub_prim2'
CRS-2676: Start of 'ora.data.acfs_gg1.acfs' on 'gghub_prim2' succeeded
CRS-2672: Attempting to start 'gghub_prim_vip1' on 'gghub_prim2'
CRS-2676: Start of 'gghub_prim_vip1' on 'gghub_prim2' succeeded

[grid@gghub_prim1 ~]$ crsctl status resource $APPVIP


NAME=gghub_prim_vip1
TYPE=app.appviptypex2.type
TARGET=ONLINE
STATE=ONLINE on gghub_prim2

[grid@gghub_prim1 ~]$ crsctl relocate resource $APPVIP -f


CRS-2673: Attempting to stop 'gghub_prim_vip1' on 'gghub_prim2'
CRS-2677: Stop of 'gghub_prim_vip1' on 'gghub_prim2' succeeded
CRS-2673: Attempting to stop 'ora.data.acfs_gg1.acfs' on 'gghub_prim2'
CRS-2677: Stop of 'ora.data.acfs_gg1.acfs' on 'gghub_prim2' succeeded
CRS-2672: Attempting to start 'ora.data.acfs_gg1.acfs' on 'gghub_prim1'
CRS-2676: Start of 'ora.data.acfs_gg1.acfs' on 'gghub_prim1' succeeded
CRS-2672: Attempting to start 'gghub_prim_vip1' on 'gghub_prim1'
CRS-2676: Start of 'gghub_prim_vip1' on 'gghub_prim1' succeeded
```

6. As the `grid` OS user on the first GGHub node, check the status of the ACFS file system:

```
[grid@gghub_prim1 ~]$ srvctl status filesystem -volume ACFS_GG1 -diskgroup
DATA

ACFS file system /mnt/acfs_gg1 is mounted on nodes gghub_prim1
```

Step 3.3.6 – Create the SSH Daemon CRS Resource

ACFS replication uses secure shell (`ssh`) to communicate between the primary and standby file systems using the virtual IP addresses that were previously created. When a server is rebooted, the `ssh` daemon is started before the VIP CRS resource, preventing access to the cluster using VIP. The following instructions create an `ssh` restart CRS resource that will restart the `ssh` daemon after the virtual IP resource is started. A separate `ssh` restart CRS resource is needed for each replicated file system.

As the `grid` OS user on all GGHUB nodes, copy the CRS action script to restart the `ssh` daemon. Place the script in the same location on all primary and standby GGHUB nodes:

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ unzip /u01/oracle/stage/gghub_scripts_<YYYYMMDD>.zip
 -d /u01/oracle/scripts/

Archive:  /u01/oracle/stage/gghub_scripts_<YYYYMMDD>.zip
  inflating: /u01/oracle/scripts/acfs_primary.scr
  inflating: /u01/oracle/scripts/acfs_standby.scr
  inflating: /u01/oracle/scripts/sshd_restart.scr
  inflating: /u01/oracle/scripts/add_acfs_primary.sh
  inflating: /u01/oracle/scripts/add_acfs_standby.sh
  inflating: /u01/oracle/scripts/add_nginx.sh
  inflating: /u01/oracle/scripts/add_sshd_restart.sh
  inflating: /u01/oracle/scripts/reverse_proxy_settings.sh
  inflating: /u01/oracle/scripts/secureServices.py
```

As the `root` OS user on the first GGHUB node, create the CRS resource using the following command:

```
[opc@gghub_prim1 ~]$ sudo su -
[root@gghub_prim1 ~]# sh /u01/oracle/scripts/add_sshd_restart.sh

Application VIP Name: gghub_prim_vip
```

As the `grid` OS user on the first GGHUB node, start and test the CRS resource:

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ crsctl stat res sshd_restart
NAME=sshd_restart
TYPE=cluster_resource
TARGET=OFFLINE
STATE=OFFLINE

[grid@gghub_prim1 ~]$ crsctl start res sshd_restart

CRS-2672: Attempting to start 'sshd_restart' on 'gghub_prim1'
CRS-2676: Start of 'sshd_restart' on 'gghub_prim1' succeeded
```

```
[grid@gghub_prim1 ~]$ cat /tmp/sshd_restarted
STARTED

[grid@gghubtest1 ~]$ crsctl stop res sshd_restart


CRS-2673: Attempting to stop 'sshd_restart' on 'gghub_prim1'
CRS-2677: Stop of 'sshd_restart' on 'gghub_prim1' succeeded

[grid@gghub1 ~]$ cat /tmp/sshd_restarted
STOPPED

[grid@gghub1 ~]$ crsctl start res sshd_restart


CRS-2672: Attempting to start 'sshd_restart' on 'gghub_prim1'
CRS-2676: Start of 'sshd_restart' on 'gghub_prim1' succeeded

[grid@gghub1 ~]$ crsctl stat res sshd_restart


NAME=sshd_restart
TYPE=cluster_resource
TARGET=ONLINE
STATE=ONLINE on gghub_prim1
```

Step 3.3.7 – Enable ACFS Replication

ACFS snapshot-based replication uses `openssh` to transfer the snapshots from between the primary and standby hosts using the designated replication user, which is commonly the `grid` user.

1. As the `grid` OS user in the primary and standby hub systems, follow the instructions provided in Configuring ssh for Use With Oracle ACFS Replication to configure the ssh connectivity between the primary and standby nodes.
2. As the `grid` OS user on all primary and standby GGHub nodes, use `ssh` to test connectivity between all primary to standby nodes, and in the reverse direction using ssh as the replication user:

```
# On the Primary GGhub
[grid@gghub_prim1 ~]$ ssh gghub_stby_vip1.frankfurt.goldengate.com hostname
gghub_stby1

[grid@gghub_prim2 ~]$ ssh gghub_stby_vip1.frankfurt.goldengate.com hostname
gghub_stby1

# On the Standby GGhub
[grid@gghub_stby1 ~]$ ssh gghub_prim_vip1.frankfurt.goldengate.com hostname
gghub_prim1

[grid@gghub_stby2 ~]$ ssh gghub_prim_vip1.frankfurt.goldengate.com hostname
gghub_prim1
```

3. As the `grid` OS user on the primary and standby GGHub nodes where ACFS is mounted, use `acfsutil` to test connectivity between the primary and the standby nodes:

```
# On the Primary GGhub

[grid@gghub_prim1 ~]$ srvctl status filesystem -volume ACFS_GG1 -diskgroup
DATA
ACFS file system /mnt/acfs_gg1 is mounted on nodes gghub_prim1

[grid@gghub_prim1 ~]$ acfsutil repl info -c -u
 grid gghub_prim_vip1.frankfurt.goldengate.com
gghub_stby_vip1.frankfurt.goldengate.com
 /mnt/acfs_gg1
A valid 'ssh' connection was detected for standby node
 gghub_prim_vip1.frankfurt.goldengate.com as user grid.
A valid 'ssh' connection was detected for standby node
 gghub_stby_vip1.frankfurt.goldengate.com as user grid.

# On the Standby GGhub

[grid@gghub_stby1 ~]$ srvctl status filesystem -volume ACFS_GG1 -diskgroup
DATA
ACFS file system /mnt/acfs_gg1 is mounted on nodes gghub_stby1

[grid@gghub_stby1 ~]$ acfsutil repl info -c -u grid
 gghub_prim_vip1.frankfurt.goldengate.com
gghub_stby_vip1.frankfurt.goldengate.com
 /mnt/acfs_gg
A valid 'ssh' connection was detected for standby node
 gghub_prim_vip1.frankfurt.goldengate.com as user grid.
A valid 'ssh' connection was detected for standby node
 gghub_stby_vip1.frankfurt.goldengate.com as user grid.
```

4. If the `acfsutil` command is executed from a GGHub node where ACFS is not mounted, the error ACFS-05518 will be shown as expected. Use `srvctl status filesytem` to find the GGHub where ACFS is mounted and re-execute the command:

```
[grid@gghub_prim1 ~]$ acfsutil repl info -c -u grid
 gghub_stby_vip1.frankfurt.goldengate.com
gghub_stby_vip1.frankfurt.goldengate.com
 /mnt/acfs_gg1
acfsutil repl info: ACFS-05518: /mnt/acfs_gg1 is not an ACFS mount point

[grid@gghub_prim1 ~]$ srvctl status filesystem -volume ACFS_GG1 -diskgroup
DATA
ACFS file system /mnt/acfs_gg1 is mounted on nodes gghub_prim2

[grid@gghub_prim1 ~]$ ssh gghub_prim2
[grid@gghub_prim2 ~]$ acfsutil repl info -c -u grid
 gghub_prim_vip1.frankfurt.goldengate.com
gghub_stby_vip1.frankfurt.goldengate.com
 /mnt/acfs_gg1
A valid 'ssh' connection was detected for standby node
 gghub_prim_vip1.frankfurt.goldengate.com as user grid.
```

```
A valid 'ssh' connection was detected for standby node
 gghub_stby_vip1.frankfurt.goldengate.com as user grid.
```

> **Note:**
>
> Make sure the connectivity is verified between all primary nodes to all standby nodes, as well as in the opposite direction. Only continue when there are no errors with any of the connection tests.

5. As the `grid` OS user on the standby GGHub node where ACFS is currently mounted, initialize ACFS replication:

```
[grid@gghub_stby1 ~]$ srvctl status filesystem -volume ACFS_GG1 -diskgroup
DATA

ACFS file system /mnt/acfs_gg1 is mounted on nodes gghub_stby1

[grid@gghub_stby1 ~]$ /sbin/acfsutil repl init standby -u grid /mnt/
acfs_gg1
```

6. As the `grid` OS user on the primary GGHub node where ACFS is currently mounted, initialize ACFS replication:

```
[grid@gghub_prim1 ~]$ srvctl status filesystem -volume ACFS_GG1 -diskgroup
DATA

ACFS file system /mnt/acfs_gg is mounted on nodes gghub_prim1

[grid@gghub_prim1 ~]$ /sbin/acfsutil repl init primary -C -p
 grid@gghub_prim_vip1.frankfurt.goldengate.com -s
 grid@gghub_stby_vip1.frankfurt.goldengate.com -m /mnt/acfs_gg1 /mnt/
acfs_gg1
```

7. As the `grid` OS user on the primary and standby GGHub nodes, monitor the initialization progress, when the status changes to “Send Completed” it means the initial primary file system copy has finished and the primary file system is now being replicated to the standby host:

```
# On the Primary GGhub

[grid@gghub_prim1 ~]$ /sbin/acfsutil repl info -c -v /mnt/acfs_gg1 | grep -
i Status

Status:                               Send Completed

# On the Standby GGhub

[grid@gghub_prim1 ~]$ /sbin/acfsutil repl info -c -v /mnt/acfs_gg1 | grep -
i Status

Status:                               Receive Completed
```

8. As the `grid` OS user on the primary and standby GGHub nodes, verify and monitor the ACFS replicated file system:

```
# On the Primary GGhub

[grid@gghub_prim1 ~]$ acfsutil repl util verifystandby /mnt/acfs_gg1

verifystandby returned: 0

# On the Standby GGhub


[grid@gghubtest31 ~]$ acfsutil repl util verifyprimary /mnt/acfs_gg1

verifyprimary returned: 0
```

> **Note:**
>
> Both commands will return a value of 0 (zero) if there are no problems detected. If a non-zero value is returned, refer to Troubleshooting ACFS Replication for monitoring, diagnosing, and resolving common issues with ACFS Replication before continuing.

9. As the `grid` OS user on the primary GGHub node, use the following command to monitor the status of the ACFS replication:

```
[grid@gghub_prim1 ~]$ /sbin/acfsutil repl info -c -v /mnt/acfs_gg1


Site:                                 Primary
Primary hostname:
 gghub_prim_vip1.frankfurt.goldengate.com
Primary path:                         /mnt/acfs_gg1
Primary status:                       Running
Background Resources:                 Active

Standby connect string:
 grid@gghub_stby_vip1.frankfurt.goldengate.com
Standby path:                         /mnt/acfs_gg1
Replication interval:                 0 days, 0 hours, 0 minutes, 0 seconds
Sending primary as of:                Fri May 05 12:37:02 2023
Status:                               Send Completed
Lag Time:                             00:00:00
Retries made:                         0
Last send started at:                 Fri May 05 12:37:02 2023
Last send completed at:               Fri May 05 12:37:12 2023
Elapsed time for last send:           0 days, 0 hours, 0 minutes, 10 seconds
Next send starts at:                  now
Replicated tags:
Data transfer compression:            Off
ssh strict host key checking:         On
Debug log level:                      3
Replication ID:                       0x4d7d34a
```

10. As the `grid` OS user on the standby GGHub node where ACFS is currently mounted, use the following command to monitor the status of the ACFS replication:

```
[grid@gghub_stby1 ~]$ /sbin/acfsutil repl info -c -v /mnt/acfs_gg1


Site:                                 Standby
Primary hostname:
 gghub_prim_vip1.frankfurt.goldengate.com
Primary path:                         /mnt/acfs_gg1

Standby connect string:
 grid@gghub_stby_vip1.frankfurt.goldengate.com
Standby path:                         /mnt/acfs_gg1
Replication interval:                 0 days, 0 hours, 0 minutes, 0 seconds
Last sync time with primary:          Fri May 05 12:37:02 2023
Receiving primary as of:              Fri May 05 12:37:02 2023
Status:                               Receive Completed
Last receive started at:              Fri May 05 12:37:02 2023
Last receive completed at:            Fri May 05 12:37:07 2023
Elapsed time for last receive:        0 days, 0 hours, 0 minutes, 5 seconds
Data transfer compression:            Off
ssh strict host key checking:         On
Debug log level:                      3
Replication ID:                       0x4d7d34a
```

Step 3.3.8 – Create the ACFS Replication CRS Action Scripts

To determine the health of the ACFS primary and standby file systems, CRS action scripts are used. At predefined intervals the action scripts report the health of the file systems into the CRS trace file `crsd_scriptagent_grid.trc`, located in the Grid Infrastructure trace file directory `/u01/app/grid/diag/crs/<node_name>/crs/trace` on each of the primary and standby file system of the GGhub nodes.

On both, the primary and standby file system clusters, there are two scripts required. One to monitor the local primary file system, and if the remote standby file system is available, and one to monitor the local standby file system and check remote primary file systems' availability. Example scripts are provided to implement the ACFS monitoring, but you must edit them to suit your environment.

Each replicated file system will need its own `acfs_primary` and `acfs_standby` action scripts.

Step 3.3.8.1 - Action Script acfs_primary.scr

The `acfs_primary` CRS resource checks whether the current ACFS mount is a primary file system and confirms that the standby file system is accessible and receiving replicated data. The resource is used to automatically determine if Oracle GoldenGate can start processes on the primary Oracle GoldenGate hub. If the standby file system is not accessible by the primary, the example script makes multiple attempts to verify the standby file system.

The `acfs_primary` CRS resource runs on both, the primary and standby hosts, but only returns success when the current file system is the primary file system, and the standby file system is accessible. The script must be placed in the same location on all primary and standby file system nodes.

The following parameters use suggested default settings, which should be tested before changing their values:

- `MOUNT_POINT=/mnt/acfs_gg1`
  `# The replicated ACFS mount point`
- `PATH_NAME=$MOUNT_POINT/status/acfs_primary`
  `# Must be unique from other mount files`
- `ATTEMPTS=3`
  `# Number of attempts to check the remote standby file system`
- `INTERVAL=10`
  `# Number of seconds between each attempt`

As the `grid` OS user on all primary and standby GGHUB nodes, edit the `acfs_primary.scr` script to match the environment:

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ vi /u01/oracle/scripts/acfs_primary.scr
```

As the `oracle` OS user on the primary GGhub node where ACFS is currently mounted, run the following commands to create the status directory:

```
[opc@gghub_prim1 ~]$ sudo su - oracle
[oracle@gghub_prim1 ~]$ mkdir /mnt/acfs_gg1/status
[oracle@gghub_prim1 ~]$ chmod g+w /mnt/acfs_gg1/status
```

As the `grid` OS user on the primary and standby GGHub node where ACFS is currently mounted, run the following command to register the `acfs_primary` action script for monitoring the primary and standby file system:

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ sh /u01/oracle/scripts/add_acfs_primary.sh

################################################################################
##
List of ACFS resources:
ora.data.acfs_gg1.acfs
################################################################################
##
ACFS resource name: <ora.data.acfs_gg1.acfs>
```

As the `grid` OS user on the primary GGHub node where ACFS is currently mounted, start and check the status of the `acfs_primary` resource:

```
[grid@gghub_prim1 ~]$ crsctl start resource acfs_primary


CRS-2672: Attempting to start 'acfs_primary' on 'gghub_prim1'
CRS-2676: Start of 'acfs_primary' on 'gghub_prim1' succeeded

[grid@gghub_prim1 ~]$ crsctl stat resource acfs_primary

NAME=acfs_primary
TYPE=cluster_resource
TARGET=ONLINE
STATE=ONLINE on gghub_prim1
```

```
[grid@gghub_prim1 ~]$ grep acfs_primary
 /u01/app/grid/diag/crs/`hostname`/crs/trace/crsd_scriptagent_grid.trc
 |grep check

2023-05-05 12:57:40.372 :CLSDYNAM:2725328640: [acfs_primary]{1:33562:34377}
 [check] Executing action script:
 /u01/oracle/scripts/acfs_primary.scr[check]
2023-05-05 12:57:42.376 :CLSDYNAM:2725328640: [acfs_primary]{1:33562:34377}
 [check] SUCCESS: STANDBY file system /mnt/acfs_gg1 is ONLINE
```

As the `grid` OS user on the standby GGHub node where ACFS is currently mounted, start and check the status of the `acfs_primary` resource. This step should fail because `acfs_primary` should **ONLY** be online on the primary GGhub:

```
[grid@gghub_stby1 ~]$ crsctl start res acfs_primary -n `hostname`


CRS-2672: Attempting to start 'acfs_primary' on 'gghub_stby1'
CRS-2674: Start of 'acfs_primary' on 'gghub_stby1' succeeded
CRS-2679: Attempting to clean 'acfs_primary' on 'gghub_stby1'
CRS-2681: Clean of 'acfs_primary' on 'gghub_stby1' succeeded
CRS-4000: Command Start failed, or completed with errors.

[grid@gghub_stby1 ~]$ crsctl stat res acfs_primary

NAME=acfs_primary
TYPE=cluster_resource
TARGET=ONLINE
STATE=OFFLINE


[grid@gghub_stby1 trace]$ grep acfs_primary
 /u01/app/grid/diag/crs/`hostname`/crs/trace/crsd_scriptagent_grid.trc
 |grep check

2023-05-05 13:09:53.343 :CLSDYNAM:3598239488: [acfs_primary]{1:8532:2106}
 [check] Executing action script: /u01/oracle/scripts/acfs_primary.scr[check]
2023-05-05 13:09:53.394 :CLSDYNAM:3598239488: [acfs_primary]{1:8532:2106}
 [check] Detected local standby file system
2023-05-05 13:09:53.493 :CLSDYNAM:1626130176: [acfs_primary]{1:8532:2106}
 [clean] Clean/Abort -- Stopping ACFS file system type checking...
```

> **Note:**
>
> The status of the `acfs_primary` resources will only be `ONLINE` if the ACFS file system is the primary file system. When starting the resources on a node which is not currently on the primary cluster an error will be reported because the resource fails due to being the standby file system. This error can be ignored. The resource will be in `OFFLINE` status on the ACFS standby cluster.

Step 3.3.8.2 - Action Script acfs_standby.scr

The `acfs_standby` resource checks that the local file system is a standby file system and verifies the remote primary file system status. If the primary file system fails verification multiple times (controlled by the action script variables), a warning is output to the CRS trace file `crsd_scriptagent_grid.trc` located in the Grid Infrastructure trace file directory `/u01/app/grid/diag/crs/<node_name>/crs/trace`.

This resource runs on both the primary and standby hosts, but only returns success when the current file system is the standby file system, and the primary file system is accessible.

The following parameters use suggested default settings, which should be tested before changing their values.

- `MOUNT_POINT=/mnt/acfs_gg`
  `# This is the replicated ACFS mount point`
- `ATTEMPTS=3`
  `# Number of tries to check the remote primary file system`
- `INTERVAL=10`
  `# Number of seconds between each attempt`

As the `grid` OS user on all primary and standby GGHUB nodes, edit the `acfs_standby.scr` script to match the environment:

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ vi /u01/oracle/scripts/acfs_standby.scr
```

As the `grid` OS user on the primary GGHUB node where ACFS is currently mounted, run the following command to register the `acfs_standby` action script for monitoring the primary and standby file system:

```
[grid@gghub_prim1 ~]$ crsctl stat res -w "TYPE co appvip"
 |grep NAME

NAME=gghub_prim_vip

[grid@gghub_prim1 ~]$ vi /u01/oracle/scripts/add_acfs_standby.sh


crsctl add resource acfs_standby \
 -type cluster_resource \
 -attr "ACTION_SCRIPT=/u01/oracle/scripts/acfs_standby.scr, \
   CHECK_INTERVAL=150, \
   CHECK_TIMEOUT=140, \
   START_DEPENDENCIES='hard(ora.data.acfs_gg1.acfs,gghub_prim_vip)
 pullup:always(ora.data.acfs_gg1.acfs,gghub_prim_vip)', \
   STOP_DEPENDENCIES='hard(ora.data.acfs_gg1.acfs,gghub_prim_vip)' \
   OFFLINE_CHECK_INTERVAL=300, \
   RESTART_ATTEMPTS=0, \
   INSTANCE_FAILOVER=0"

[grid@gghub_prim1 ~]$ sh /u01/oracle/scripts/add_acfs_standby.sh
```

As the `grid` OS user on the primary GGHUB node where ACFS is currently mounted, start and check the status of the `acfs_standby` resource:

```
[grid@gghub_prim1 ~]$ crsctl start res acfs_standby

CRS-2672: Attempting to start 'acfs_standby' on 'gghub_prim1'
CRS-2676: Start of 'acfs_standby' on 'gghub_prim1' succeeded

[grid@gghub_prim1 ~]$ grep acfs_standby
 /u01/app/grid/diag/crs/`hostname`/crs/trace/crsd_scriptagent_grid.trc
 |egrep 'check|INFO'

2023-05-05 13:22:09.612 :CLSDYNAM:2725328640: [acfs_standby]{1:33562:34709}
 [start] acfs_standby.scr starting to check ACFS remote primary at
 /mnt/acfs_gg1
2023-05-05 13:22:09.612 :CLSDYNAM:2725328640: [acfs_standby]{1:33562:34709}
 [check] Executing action script: /u01/oracle/scripts/acfs_standby.scr[check]
2023-05-05 13:22:09.663 :CLSDYNAM:2725328640: [acfs_standby]{1:33562:34709}
 [check] Local PRIMARY file system /mnt/acfs_gg1
```

As the `grid` OS user on the standby GGHUB node where ACFS is currently mounted, run the following command to register the `acfs_standby` action script for monitoring the primary and standby file system:

```
[grid@gghub_stby1 ~]$ crsctl stat res -w "TYPE co appvip"
 |grep NAME

NAME=gghub_stby_vip

[grid@gghub_stby1 ~]$ vi /u01/oracle/scripts/add_acfs_standby.sh

crsctl add resource acfs_standby \
 -type cluster_resource \
 -attr "ACTION_SCRIPT=/u01/oracle/scripts/acfs_standby.scr, \
   CHECK_INTERVAL=150, \
   CHECK_TIMEOUT=140, \
   START_DEPENDENCIES='hard(ora.data.acfs_gg1.acfs,gghub_stby_vip)
 pullup:always(ora.data.acfs_gg1.acfs,gghub_stby_vip)', \
   STOP_DEPENDENCIES='hard(ora.data.acfs_gg1.acfs,gghub_stby_vip)' \
   OFFLINE_CHECK_INTERVAL=300, \
   RESTART_ATTEMPTS=0, \
   INSTANCE_FAILOVER=0"

[grid@gghub_stby1 ~]$ sh /u01/oracle/scripts/add_acfs_standby.sh
```

As the `grid` OS user on the primary GGHUB node where ACFS is currently mounted, start and check the status of the `acfs_standby` resource:

```
[grid@gghub_stby1 ~]$ crsctl start res acfs_standby

CRS-2672: Attempting to start 'acfs_standby' on 'gghub_stby1'
CRS-2676: Start of 'acfs_standby' on 'gghub_stby1' succeeded

[grid@gghub_stby1 ~]$ grep acfs_standby
```

```
 /u01/app/grid/diag/crs/`hostname`/crs/trace/crsd_scriptagent_grid.trc
 |egrep 'check|INFO'
2023-05-05 13:25:20.699 :CLSDYNAM:1427187456: [acfs_standby]{1:8532:2281}
 [check] SUCCESS: PRIMARY file system /mnt/acfs_gg1 is ONLINE
2023-05-05 13:25:20.699 :    AGFW:1425086208: [     INFO] {1:8532:2281}
 acfs_standby 1 1 state changed from: STARTING to: ONLINE
2023-05-05 13:25:20.699 :    AGFW:1425086208: [     INFO] {1:8532:2281}
 Started implicit monitor for [acfs_standby 1 1]
 interval=150000 delay=150000
2023-05-05 13:25:20.699 :    AGFW:1425086208: [     INFO] {1:8532:2281}
 Agent sending last reply for: RESOURCE_START[acfs_standby 1 1]
 ID 4098:8346
```

Step 3.3.9 – Test ACFS GGhub Node Relocation

It is very important to test planned and unplanned ACFS GGhub node relocations and server role transitions before configuring Oracle GoldenGate.

As the `grid` OS user on the primary and standby GGHUB nodes, run the following command to relocate ACFS between the GGhub nodes:

```
[grid@gghub_prim1 ~]$ srvctl status filesystem -volume ACFS_GG1
 -diskgroup DATA

ACFS file system /mnt/acfs_gg1 is mounted on nodes gghub_prim1

[grid@gghub_prim1 ~]$ srvctl relocate filesystem -diskgroup DATA
 -volume acfs_gg1 -force

[grid@gghub_prim1 ~]$ srvctl status filesystem -volume ACFS_GG1
 -diskgroup DATA

ACFS file system /mnt/acfs_gg1 is mounted on nodes gghub_prim2
```

As the `grid` OS user on the primary and standby GGHUB nodes, verify that the file system is mounted on another node, along with the VIP, `sshd_restart`, and the two ACFS resources (`acfs_primary` and `acfs_standby`) using the following example command:

```
[grid@gghub_prim1 ~]$ crsctl stat res sshd_restart acfs_primary
 acfs_standby ora.data.acfs_gg1.acfs sshd_restart -t

--------------------------------------------------------------------------------
--
Name           Target  State        Server                   State details

--------------------------------------------------------------------------------
--
Cluster Resources
--------------------------------------------------------------------------------
--
acfs_primary
      1        ONLINE  ONLINE       gghub_prim2              STABLE
acfs_standby
      1        ONLINE  ONLINE                                STABLE
gghubfad2
```

```
      1        ONLINE  ONLINE       gghub_prim2               STABLE
ora.data.acfs_gg1.acfs
      1        ONLINE  ONLINE       gghub_prim2               mounted on /mnt/
acfs
                                                              _gg1,STABLE
sshd_restart
      1        ONLINE  ONLINE       gghub_prim2               STABLE
--------------------------------------------------------------------------------
--

[grid@gghub_stby1 ~]$ crsctl stat res sshd_restart acfs_primary acfs_standby
 ora.data.acfs_gg1.acfs sshd_restart -t


--------------------------------------------------------------------------------
--
Name           Target  State        Server                   State details
--------------------------------------------------------------------------------
--
Cluster Resources
--------------------------------------------------------------------------------
--
acfs_primary
      1        ONLINE  OFFLINE                                STABLE
acfs_standby
      1        ONLINE  ONLINE       gghub_stby2               STABLE
ora.data.acfs_gg1.acfs
      1        ONLINE  ONLINE       gghub_stby2               mounted on /mnt/
acfs
                                                              _gg1,STABLE
sshd_restart
      1        ONLINE  ONLINE       gghub_stby2               STABLE
--------------------------------------------------------------------------------
--
```

Step 3.3.10 – Test ACFS Switchover Between the Primary and Standby GGhub

As the `grid` OS user on the standby GGHUB node, run the following command to issue an ACFS switchover (role reversal) between the primary and standby GGhub:

```
[grid@gghub_stby2 ~]$ crsctl stat res ora.data.acfs_gg1.acfs

NAME=ora.data.acfs_gg.acfs
TYPE=ora.acfs_cluster.type
TARGET=ONLINE
STATE=ONLINE on gghub_stby2

[grid@gghub_stby2 ~]$ acfsutil repl failover /mnt/acfs_gg1


[grid@gghub_stby2 ~]$ /sbin/acfsutil repl info -c -v /mnt/acfs_gg1


Site:                                 Primary
Primary hostname:                     gghub_stby_vip.frankfurt.goldengate.com
Primary path:                         /mnt/acfs_gg1
```

```
Primary status:                         Running
Background Resources:                   Active

Standby connect string:                 gghub_prim_vip.frankfurt.goldengate.com
Standby path:                           /mnt/acfs_gg1
Replication interval:                   0 days, 0 hours, 0 minutes, 0 seconds
Sending primary as of:                  Fri May 05 13:51:37 2023
Status:                                 Send Completed
Lag Time:                               00:00:00
Retries made:                           0
Last send started at:                   Fri May 05 13:51:37 2023
Last send completed at:                 Fri May 05 13:51:48 2023
Elapsed time for last send:             0 days, 0 hours, 0 minutes, 11 seconds
Next send starts at:                    now
Replicated tags:
Data transfer compression:              Off
ssh strict host key checking:           On
Debug log level:                        3
Replication ID:                         0x4d7d34a
```

As the `grid` OS user on the new standby GGHUB node (old primary), run the following command to issue an ACFS switchover (role reversal) between the primary and standby GGhub. This step is optional but recommended to return the sites to the original role:

```
[grid@gghub_prim2 ~]$ crsctl stat res ora.data.acfs_gg1.acfs

NAME=ora.data.acfs_gg1.acfs
TYPE=ora.acfs_cluster.type
TARGET=ONLINE
STATE=ONLINE on gghub_prim2


[grid@gghub_prim2 ~]$ /sbin/acfsutil repl info -c -v /mnt/acfs_gg1 |grep Site
Site:                                   Standby

[grid@gghub_prim2 ~]$ acfsutil repl failover /mnt/acfs_gg1


[grid@gghub_prim2 ~]$ /sbin/acfsutil repl info -c -v /mnt/acfs_gg1 |grep Site

Site:                                   Primary
```

Step 3.4 - Create the Oracle GoldenGate Deployment

Once the Oracle GoldenGate software has been installed in GGHub, the next step is to create a response file to create the GoldenGate deployment using the Oracle GoldenGate Configuration Assistant.

Due the unified build feature introduced in Oracle GoldenGate 21c, a single deployment can now manage Extract and Replicat processes that attach to different Oracle Database versions. Each deployment is created with an Administration Server and (optionally) Performance Metrics Server. If the GoldenGate trail files don't need to be transferred to another hub or GoldenGate environment, there is no need to create a Distribution or Receiver Server.

There are two limitations that currently exist with Oracle GoldenGate and XAG:

- A Service Manager that is registered with XAG can only manage a single deployment. If multiple deployments are required, each deployment must use their own Service Manager. Oracle GoldenGate release 21c simplifies this requirement because it uses a single deployment to support Extract and Replicat processes connecting to different versions of the Oracle Database.
- Each Service Manager registered with XAG must belong to separate `OGG_HOME` software installation directories. Instead of installing Oracle GoldenGate multiple times, the recommended approach is to install Oracle GoldenGate one time, and then create a symbolic link for each Service Manager `OGG_HOME`. The symbolic link and `OGG_HOME` environment variable must be configured **before** running the Oracle GoldenGate Configuration Assistant on all Oracle RAC nodes.

1. Create a Response File

   For a silent configuration, please copy the following example file and paste it into any location the oracle user can access. Edit the following values appropriately:

   - `CONFIGURATION_OPTION`
   - `DEPLOYMENT_NAME`
   - `ADMINISTRATOR_USER`
   - `SERVICEMANAGER_DEPLOYMENT_HOME`
   - `OGG_SOFTWARE_HOME`
   - `OGG_DEPLOYMENT_HOME`
   - `ENV_TNS_ADMIN`
   - `OGG_SCHEMA`

   Example Response File (`oggca.rsp`):

   As the `oracle` OS user on the primary GGHUB node where ACFS is currently mounted, create and edit the response file `oggca.rsp` to create the Oracle GoldenGate deployment:

```
[opc@gghub_prim1 ~]$ sudo su - oracle
[oracle@gghub_prim1 ~]$ vi /u01/oracle/stage/oggca.rsp

oracle.install.responseFileVersion=/oracle/install/
rspfmt_oggca_response_schema_v21_1_0
CONFIGURATION_OPTION=ADD
DEPLOYMENT_NAME=gghub1
ADMINISTRATOR_USER=oggadmin
ADMINISTRATOR_PASSWORD=<password_for_oggadmin>
SERVICEMANAGER_DEPLOYMENT_HOME=/mnt/acfs_gg1/deployments/ggsm01
HOST_SERVICEMANAGER=localhost
PORT_SERVICEMANAGER=9100
SECURITY_ENABLED=false
STRONG_PWD_POLICY_ENABLED=true
CREATE_NEW_SERVICEMANAGER=true
REGISTER_SERVICEMANAGER_AS_A_SERVICE=false
INTEGRATE_SERVICEMANAGER_WITH_XAG=true
EXISTING_SERVICEMANAGER_IS_XAG_ENABLED=false
OGG_SOFTWARE_HOME=/u01/app/oracle/goldengate/gg21c
OGG_DEPLOYMENT_HOME=/mnt/acfs_gg1/deployments/gg01
ENV_LD_LIBRARY_PATH=${OGG_HOME}/lib/instantclient:${OGG_HOME}/lib
```

```
ENV_TNS_ADMIN=/u01/app/oracle/goldengate/network/admin
FIPS_ENABLED=false
SHARDING_ENABLED=false
ADMINISTRATION_SERVER_ENABLED=true
PORT_ADMINSRVR=9101
DISTRIBUTION_SERVER_ENABLED=true
PORT_DISTSRVR=9102
NON_SECURE_DISTSRVR_CONNECTS_TO_SECURE_RCVRSRVR=false
RECEIVER_SERVER_ENABLED=true
PORT_RCVRSRVR=9103
METRICS_SERVER_ENABLED=true
METRICS_SERVER_IS_CRITICAL=false
PORT_PMSRVR=9104
UDP_PORT_PMSRVR=9105
PMSRVR_DATASTORE_TYPE=BDB
PMSRVR_DATASTORE_HOME=/u01/app/oracle/goldengate/datastores/gghub1
OGG_SCHEMA=ggadmin
```

2. Create the Oracle GoldenGate Deployment

   As the `oracle` OS user on the primary GGHUB node where ACFS is currently mounted, run `oggca.sh` to create the GoldenGate deployment:

```
[opc@gghub_prim1 ~]$ sudo su - oracle
[oracle@gghub_prim1 ~]$ export OGG_HOME=/u01/app/oracle/goldengate/gg21c
[oracle@gghub_prim1 ~]$ $OGG_HOME/bin/oggca.sh -silent
 -responseFile /u01/oracle/stage/oggca.rsp

Successfully Setup Software.
```

3. Create the Oracle GoldenGate Datastores and TNS_ADMIN Directories

   As the `oracle` OS user on all GGHUB nodes of the primary and standby systems, run the following commands to create the Oracle GoldenGate Datastores and `TNS_ADMIN` directories:

```
[opc@gghub_prim1 ~]$ sudo su - oracle
[oracle@gghub_prim1 ~]$ mkdir -p /u01/app/oracle/goldengate/network/admin
[oracle@gghub_prim1 ~]$ mkdir -p /u01/app/oracle/goldengate/datastores/
gghub1
```

Step 3.5 - Configure Oracle Grid Infrastructure Agent (XAG)

The following step-by-step procedure shows how to configure Oracle Clusterware to manage GoldenGate using the Oracle Grid Infrastructure Standalone Agent (XAG). Using XAG automates the ACFS file system mounting, as well as the stopping and starting of the GoldenGate deployment when relocating between Oracle GGhub nodes.

Step 3.5.1 - Install the Oracle Grid Infrastructure Standalone Agent

It is recommended to install the XAG software as a standalone agent outside the Grid Infrastructure `ORACLE_HOME`. This way, you can use the latest XAG release available, and the software can be updated without impact to the Grid Infrastructure.

Install the XAG standalone agent outside of the Oracle Grid Infrastructure home directory. XAG must be installed in the same directory on all GGhub nodes in the system where GoldenGate is installed.

As the `grid` OS user on the first GGHub node of the primary and standby systems, unzip the software and run `xagsetup.sh`:

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ unzip /u01/oracle/stage/p31215432_190000_Generic.zip
 -d /u01/oracle/stage
[grid@gghub_prim1 ~]$ /u01/oracle/stage/xag/xagsetup.sh --install
 --directory /u01/app/grid/xag --all_nodes

Installing Oracle Grid Infrastructure Agents on: gghub_prim1
Installing Oracle Grid Infrastructure Agents on: gghub_prim2
Updating XAG resources.
Successfully updated XAG resources.
```

As the `grid` OS user on all GGHUB nodes of the primary and standby systems, add the location of the newly installed XAG software to the PATH variable so that the location of `agctl` is known when the `grid` user logs on to the machine.

```
[grid@gghub_prim1 ~]$ vi ~/.bashrc

PATH=/u01/app/grid/xag/bin:$PATH:/u01/app/19.0.0.0/grid/bin; export PATH
```

> **Note:**
>
> It is essential to ensure that the XAG bin directory is specified BEFORE the Grid Infrastructure bin directory to ensure the correct `agctl` binary is found. This should be set in the grid user environment to take effect when logging on, such as in the .bashrc file when the Bash shell is in use.

Step 3.5.2 - Register Oracle Grid Infrastructure Agent on the Primary and Standby GGhubs

The following procedure shows how to configure Oracle Clusterware to manage Oracle GoldenGate using the Oracle Grid Infrastructure Standalone Agent (XAG). Using XAG automates the mounting of the shared file system as well as the stopping and starting of the Oracle GoldenGate deployment when relocating between Oracle GGhub nodes.

Oracle GoldenGate must be registered with XAG so that the deployment is started and stopped automatically when the database is started, and the file system is mounted.

To register Oracle GoldenGate Microservices Architecture with XAG, use the following command format.

```
agctl add goldengate <instance_name>
--gg_home <GoldenGate_Home>
--service_manager
--config_home <GoldenGate_SvcMgr_Config>
--var_home <GoldenGate_SvcMgr_Var Dir>
--port <port number>
--oracle_home <$OGG_HOME/lib/instantclient>
```

```
--adminuser <OGG admin user>
--user <GG instance user>
--group <GG instance group>
--file systems <CRS_resource_name>
--db_services <service_name>
--use_local_services
--attribute START_TIMEOUT=60
```

Where:

- `--gg_home` specifies the location of the GoldenGate software.
- `--service_manager` indicates this is an GoldenGate Microservices instance.
- `--config_home` specifies the GoldenGate deployment configuration home directory.
- `--var_home` specifies the GoldenGate deployment variable home directory.
- `--oracle_home` specifies the Oracle Instant Client home
- `--port` specifies the deployment Service Manager port number.
- `--adminuser` specifies the GoldenGate Microservices administrator account name.
- `--user` specifies the name of the operating system user that owns the GoldenGate deployment.
- `--group` specifies the name of the operating system group that owns the GoldenGate deployment.
- `--filesystems` specifies the CRS file system resource that must be ONLINE before the deployment is started. This will be the acfs_primary resource created in a previous step.
- `--filesystem_verify` specifies if XAG should check the existence of the directories specified by the `config_home` and `var_home` parameters. This should be set to `yes` for the active ACFS primary file system. When adding the GoldenGate instance on the standby cluster, specify `no`.
- `--filesystems_always` specifies that XAG will start the GoldenGate Service Manager on the same GGhub node as the file system CRS resources, specified by the `--filesystems` parameter.
- `--attributes` specifies that the target status of the resource is online. This is required to automatically start the GoldenGate deployment when the `acfs_primary` resource starts.

The GoldenGate deployment must be registered on the primary and standby GGHUBs where ACFS is mounted in either read-write or read-only mode.

As the `grid` OS user on the first GGHUB node of the primary and standby systems, run the following command to determine which node of the cluster the file system is mounted on:

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ crsctl stat res acfs_standby |grep STATE
STATE=ONLINE on gghub_prim1
```

Step 3.5.2.1 - Register the Primary Oracle GoldenGate Microservices Architecture with XAG

As the `root` OS user on the first node of the primary GGHUB, register Oracle GoldenGate Microservices Architecture with XAG using the following command format:

```
[opc@gghub_prim1 ~]$ sudo su - root
[root@gghub_prim1 ~]# vi /u01/oracle/scripts/add_xag_goldengate.sh

# Run as ROOT:
/u01/app/grid/xag/bin/agctl add goldengate gghub1 \
--gg_home /u01/app/oracle/goldengate/gg21c \
--service_manager \
--config_home /mnt/acfs_gg1/deployments/ggsm01/etc/conf \
--var_home /mnt/acfs_gg1/deployments/ggsm01/var \
--oracle_home /u01/app/oracle/goldengate/gg21c/lib/instantclient \
--port 9100 \
--adminuser oggadmin \
--user oracle \
--group oinstall \
--filesystems acfs_primary \
--filesystems_always yes \
--filesystem_verify yes \
--attribute TARGET_DEFAULT=online

[root@gghub_prim1 ~]# sh /u01/oracle/scripts/add_xag_goldengate.sh
Enter password for 'oggadmin' : ##########
```

As the `grid` OS user on the first node of the primary GGHUB, verify that Oracle GoldenGate Microservices Architecture is registered with XAG:

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ agctl status goldengate

Goldengate  instance 'gghub1' is not running
```

Step 3.5.2.2 - Register the Standby Oracle GoldenGate Microservices Architecture with XAG

As the `root` OS user on the first node of the standby GGHUB, register Oracle GoldenGate Microservices Architecture with XAG using the following command format:

```
[opc@gghub_stby1 ~]$ sudo su - root
[root@gghub_stby1 ~]# vi /u01/oracle/scripts/add_xag_goldengate.sh

# Run as ROOT:
/u01/app/grid/xag/bin/agctl add goldengate gghub1 \
--gg_home /u01/app/oracle/goldengate/gg21c \
--service_manager \
--config_home /mnt/acfs_gg1/deployments/ggsm01/etc/conf \
--var_home /mnt/acfs_gg1/deployments/ggsm01/var \
--oracle_home /u01/app/oracle/goldengate/gg21c/lib/instantclient \
--port 9100 --adminuser oggadmin --user oracle --group oinstall \
--filesystems acfs_primary \
--filesystems_always yes \
--filesystem_verify no \
--attribute TARGET_DEFAULT=online
```

```
[root@gghub_stby1 ~]# sh /u01/oracle/scripts/add_xag_goldengate.sh
Enter password for 'oggadmin' : ##########
```

> **Note:**
>
> When adding the GoldenGate instance on the standby cluster, specify `--filesystem_verify no`.

As the `grid` OS user on the first node of the standby GGHUB, verify that Oracle GoldenGate Microservices Architecture is registered with XAG:

```
[opc@gghub_stby1 ~]$ sudo su - grid
[grid@gghub_stby1 ~]$ agctl status goldengate

Goldengate  instance 'gghub1' is not running
```

Step 3.5.3 - Start the Oracle GoldenGate Deployment

Below is some example agctl commands used to manage the GoldenGate deployment with XAG.

As the `grid` OS user on the first node of the primary GGHUB, run the following command to start and check Oracle GoldenGate deployment:

```
[opc@gghub_prim1 ~]$ sudo su - grid

[grid@gghub_prim1 ~]$ agctl start goldengate gghub1

[grid@gghub_prim1 ~]$ agctl status goldengate
Goldengate  instance 'gghub1' is running on gghub_prim1
```

As the `grid` OS user on the first GGHUB node, run the following command to validate the configuration parameters for the Oracle GoldenGate resource:

```
[grid@gghub_prim1 ~]$ agctl config goldengate gghub1


Instance name: gghub1
Application GoldenGate location is: /u01/app/oracle/goldengate/gg21c
Goldengate MicroServices Architecture environment: yes
Goldengate Service Manager configuration directory:
 /mnt/acfs_gg1/deployments/ggsm01/etc/conf
Goldengate Service Manager var directory:
 /mnt/acfs_gg1/deployments/ggsm01/var
Service Manager Port: 9100
Goldengate Administration User: oggadmin
Autostart on DataGuard role transition to PRIMARY: no
ORACLE_HOME location is:
 /u01/app/oracle/goldengate/gg21c/lib/instantclient
File System resources needed: acfs_primary
CRS additional attributes set: TARGET_DEFAULT=online
```

For more information see Oracle Grid Infrastructure Bundled Agent.

Step 3.6 - Configure NGINX Reverse Proxy

The GoldenGate reverse proxy feature allows a single point of contact for all the GoldenGate microservices associated with a GoldenGate deployment. Without a reverse proxy, the GoldenGate deployment microservices are contacted using a URL consisting of a hostname or IP address and separate port numbers, one for each of the services. For example, to contact the Service Manager, you could use http://gghub.example.com:9100, then the Administration Server is http://gghub.example.com:9101, the second Service Manager may be accessed using http://gghub.example.com:9110, and so on.

When running Oracle GoldenGate in a High Availability (HA) configuration on Oracle Exadata Database Service with the Grid Infrastructure agent (XAG), there is a limitation preventing more than one deployment from being managed by a GoldenGate Service Manager. Because of this limitation, creating a separate virtual IP address (VIP) for each Service Manager/ deployment pair is recommended. This way, the microservices can be accessed directly using the VIP.

With a reverse proxy, port numbers are not required to connect to the microservices because they are replaced with the deployment name and the host name's VIP. For example, to connect to the console via a web browser, use the URLs:

GoldenGate Services	URL
Service Manager	https://localhost:*localPort*
Administration Server	https://localhost:*localPort*/*instance_name*/adminsrvr
Distribution Server	https://localhost:*localPort*/*instance_name*/distsrvr
Performance Metric Server	https://localhost:*localPort*/*instance_name*/pmsrvr
Receiver Server	https://localhost:*localPort*/*instance_name*/recvsrvr

> **Note:**
>
> To connect to Oracle GoldenGate in OCI, you must create a bastion (see Step 3.2) and an SSH port forwarding session (see Step 4.1). After this, you can connect to the Oracle GoldenGate Services using https://locahost:*localPort*.

A reverse proxy is mandatory to ensure easy access to microservices and enhance security and manageability.

When running multiple Service Managers, the following instructions will provide configuration using a separate VIP for each Service Manager. NGINX uses the VIP to determine which Service Manager an HTTPS connection request is routed to.

An SSL certificate is required for clients to authenticate the server they connect to through NGINX. Contact your systems administrator to follow your corporate standards to create or obtain the server certificate before proceeding. A separate certificate is required for each VIP and Service Manager pair.

> **Note:**
>
> The common name in the CA-signed certificate must match the target hostname/VIP used by NGINX.

Follow the instructions to install and configure NGINX Reverse Proxy with an SSL connection and ensure all external communication is secure.

Step 3.6.1 - Secure Deployments Requirements (Certificates)

A secure deployment involves making RESTful API calls and conveying trail data between the Distribution Server and Receiver Server, over SSL/TLS. You can use your own existing business certificate from your Certificate Authority (CA) or you might create your own certificates. Contact your systems administrator to follow your corporate standards to create or obtain the server certificate before proceeding. A separate certificate is required for each VIP and Service Manager pair.

Step 3.6.2 - Install NGINX Reverse Proxy Server

As the `root` OS user on all GGHUB nodes, set up the `yum` repository by creating the file `/etc/yum.repos.d/nginx.repo` with the following contents:

```
[opc@gghub_prim1 ~]$ sudo su -
[root@gghub_prim1 ~]# cat > /etc/yum.repos.d/nginx.repo <<EOF
[nginx-stable]
name=nginx stable repo
baseurl=http://nginx.org/packages/rhel/7/\$basearch/
gpgcheck=1
enabled=1
gpgkey=https://nginx.org/keys/nginx_signing.key
module_hotfixes=true
EOF
```

As the `root` OS user on all GGHUB nodes, run the following commands to install, enable, and start NGINX:

```
[root@gghub_prim1 ~]# yum install -y python-requests python-urllib3 nginx
[root@gghub_prim1 ~]# systemctl enable nginx
```

As the `root` OS user on all GGHUB node, disable the NGINX repository after the software has been installed:

```
[root@gghub_prim1 ~]# yum-config-manager --disable nginx-stable
```

Step 3.6.3 - Create the NGINX Configuration File

You can configure Oracle GoldenGate Microservices Architecture to use a reverse proxy. Oracle GoldenGate MA includes a script called `ReverseProxySettings` that generates a configuration file for only the NGINX reverse proxy server.

The script requires the following parameters:

- The --user parameter should mirror the GoldenGate administrator account specified with the initial deployment creation.

- The GoldenGate administrator password will be prompted.
- The reverse proxy port number specified by the --port parameter should be the default HTTPS port number (443) unless you are running multiple GoldenGate Service Managers using the same --host. In this case, specify an HTTPS port number that does not conflict with previous Service Manager reverse proxy configurations. For example, if running two Service Managers using the same hostname/VIP, the first reverse proxy configuration is created with '--port 443 --host hostvip01', and the second is created with '--port 444 --host hostvip01'. If using separate hostnames/VIPs, the two Service Manager reverse proxy configurations would be created with '--port 443 --host hostvip01' and '--port 443 --host hostvip02'.
- Lastly, the HTTP port number (9100) should match the Service Manager port number specified during the deployment creation.

Repeat this step for each additional GoldenGate Service Manager.

As the `oracle` OS user on the first GGHUB node, use the following command to create the Oracle GoldenGate NGINX configuration file:

```
[opc@gghub_prim1 ~]$ sudo su - oracle
[oracle@gghub_prim1 ~]$ export OGG_HOME=/u01/app/oracle/goldengate/gg21c
[oracle@gghub_prim1 ~]$ export PATH=$PATH:$OGG_HOME/bin
[oracle@gghub_prim1 ~]$ cd /u01/oracle/scripts
[oracle@gghub_prim1 ~]$ $OGG_HOME/lib/utl/reverseproxy/ReverseProxySettings
 --user oggadmin --port 443 --output ogg_<gghub1>.conf http://localhost:9100
 --host <VIP hostname>
Password: <oggadmin_password>
```

Step 3.6.4 - Modify NGINX Configuration Files

When multiple GoldenGate Service Managers are configured to use their IP/VIPs with the same HTTPS 443 port, some small changes are required to the NGINX reverse proxy configuration files generated in the previous step. With all Service Managers sharing the same port number, they are independently accessed using their VIP/IP specified by the `--host` parameter.

As the `oracle` OS user on the first GGHUB node, determine the deployment name managed by this Service Manager listed in the reverse proxy configuration file and change all occurrences of “_ServiceManager” by prepending the deployment name before the underscore:

```
[oracle@gghub_prim1 ~]$ cd /u01/oracle/scripts

[oracle@gghub_prim1 ~]$ grep "Upstream Servers" ogg_<gghub1>.conf

## Upstream Servers for Deployment 'gghub1'

[oracle@gghub_prim1 ~]$  sed -i 's/_ServiceManager/<gghub1>_ServiceManager/'
 ogg_<gghub1>.conf
```

Step 3.6.5 - Install the Server Certificates for NGINX

As the `root` OS user on the first GGHUB node, copy the server certificates and key files in the `/etc/nginx/ssl` directory, owned by `root` with file permissions 400 (-r--------):

```
[opc@gghub_prim1 ~]$ sudo su -
[root@gghub_prim1 ~]# mkdir /etc/nginx/ssl
[root@gghub_prim1 ~]# cp <ssl_keys> /etc/nginx/ssl/.
[root@gghub_prim1 ~]# chmod 400 /etc/nginx/ssl
[root@gghub_prim1 ~]# ll /etc/nginx/ssl

-r-------- 1 root root 2750 May 17 06:12 gghub1.chained.crt
-r-------- 1 root root 1675 May 17 06:12 gghub1.key
```

As the `oracle` OS user on the first GGHUB node, set the correct file names for the certificate and key files for each reverse proxy configuration file:

```
[root@gghub_prim1 ~]$ vi /u01/oracle/scripts/ogg_<gghub1>.conf

# Before
    ssl_certificate         /etc/nginx/ogg.pem;
    ssl_certificate_key     /etc/nginx/ogg.pem;

# After
    ssl_certificate         /etc/nginx/ssl/gghub1.chained.crt;
    ssl_certificate_key     /etc/nginx/ssl/gghub1.key;
```

When using CA-signed certificates, the certificate named with the ssl_certificate NGINX parameter must include the 1) CA signed, 2) intermediate, and 3) root certificates in a single file. The order is significant; otherwise, NGINX fails to start and displays the error message:

```
(SSL: error:0B080074:x509 certificate routines:
  X509_check_private_key:key values mismatch)
```

The root and intermediate certificates can be downloaded from the CA-signed certificate provider.

As the `root` OS user on the first GGHUB node, generate the SSL certificate single file by using the following example command:

```
[root@gghub_prim1 ~]# cd /etc/nginx/ssl
[root@gghub_prim1 ~]# cat CA_signed_cert.crt
 intermediate.crt root.crt > gghub1.chained.crt
```

The `ssl_certificate_key` file is generated when creating the Certificate Signing Request (CSR), which is required when requesting a CA-signed certificate.

Step 3.6.6 - Install the NGINX Configuration File

As the `root` OS user on the first GGhub node, copy the deployment configuration file to `/etc/nginx/conf.d` directory and remove the default configuration file:

```
[root@gghub_prim1 ~]# cp /u01/oracle/scripts/ogg_<gghub1>.conf
 /etc/nginx/conf.d
[root@gghub_prim1 ~]# rm /etc/nginx/conf.d/default.conf
```

As the `root` OS user on the first GGHUB node, validate the NGINX configuration file. If there are errors in the file, they will be reported with the following command:

```
[root@gghub_prim1 ~]# nginx -t

nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginxconf test is successful
```

As the `root` OS user on the first GGHUB node, restart NGINX to load the new configuration:

```
[root@gghub_prim1 ~]# systemctl restart nginx
```

Step 3.6.7 - Test GoldenGate Microservices Connectivity

As the `root` OS user on the first GGHUB node, create a curl configuration file (`access.cfg`) that contains the deployment user name and password:

```
[root@gghub_prim1 ~]# vi access.cfg
user = "oggadmin:<password>"

[root@gghub_prim1 ~]# curl -svf
 -K access.cfg https://<VIP hostname>:<port#>/services/v2/config/health
 -XGET && echo -e "\n*** Success"

Sample output:
* About to connect() to gghub_prim_vip.frankfurt.goldengate.com port 443 (#0)
*   Trying 10.40.0.75...
* Connected to gghub_prim_vip.frankfurt.goldengate.com (10.40.0.75) port 443
(#0)
* Initializing NSS with certpath: sql:/etc/pki/nssdb
*   CAfile: /etc/pki/tls/certs/ca-bundle.crt
  CApath: none
* skipping SSL peer certificate verification
* NSS: client certificate not found (nickname not specified)
* SSL connection using TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
* Server certificate:
*       subject: CN=gghub_prim_vip.frankfurt.goldengate.com,OU=Oracle MAA,
O=Oracle,L=Frankfurt,ST=Frankfurt,C=GE
*       start date: Jul 27 15:59:00 2023 GMT
*       expire date: Jul 26 15:59:00 2024 GMT
*       common name: gghub_prim_vip.frankfurt.goldengate.com
*       issuer: OID.2.5.29.19=CA:true,
CN=gghub_prim_vip.frankfurt.goldengate.com,OU=Oracle
MAA,O=Oracle,L=Frankfurt,C=EU
* Server auth using Basic with user 'oggadmin'
> GET /services/v2/config/health HTTP/1.1
> Authorization: Basic b2dnYWRtaW46V0VsY29tZTEyM19fXw==
> User-Agent: curl/7.29.0
> Host: gghub_prim_vip.frankfurt.goldengate.com
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: nginx/1.24.0
< Date: Thu, 27 Jul 2023 16:25:26 GMT
< Content-Type: application/json
```

```
< Content-Length: 941
< Connection: keep-alive
< Set-Cookie:

ogg.sca.mS+pRfBERzqE+RTFZPPoVw=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOi
Jv
Z2cuc2NhIiwiZXhwIjozNjAwLCJ0eXAiOiJ4LVNDQS1BdXRob3JpemF0aW9uIiwic3ViIjoib2dnYW
Rta
W4iLCJhdWQiOiJvZ2cuc2NhIiwiaWF0IjoxNjkwNDc1MTI2LCJob3N0IjoiZ2dodWJsYV92aXAubG9
uZG
9uLmdvbGRlbmdhdGUuY29tIiwicm9sZSI6IlNlY3VyaXR5IiwiYXV0aFR5cGUiOiJCYXNpYyIsImNy
ZWQ
iOiJFd3VqV0hOdzlGWDNHai9FN1RYU3A1N1dVRjBheUd4OFpCUTdiZDlKOU9RPSIsInNlcnZlcklEI
joi
ZmFkNWVkN2MtZThlYi00YmE2LTg4Y2EtNmQxYjk3ZjdiMGQ3IiwiZGVwbG95bWVudElEIjoiOTkyZm
E5N
DUtZjA0NC00NzNhLTg0ZjktMTRjNTY0ZjNlODU3In0=.knACABXPmZE4BEyux7lZQ5GnrSCCh4x1zB
VBL
aX3Flo=; Domain=gghub_prim_vip.frankfurt.goldengate.com; Path=/; HttpOnly;
Secure;
 SameSite=strict
< Set-Cookie:

ogg.csrf.mS+pRfBERzqE+RTFZPPoVw=1ae439e625798ee02f8f7498438f27c7bad036b270d6bf
c9
5aee60fcee111d35ea7e8dc5fb5d61a38d49cac51ca53ed9307f9cbe08fab812181cf163a743bf
c7;
 Domain=gghub_prim_vip.frankfurt.goldengate.com; Path=/; Secure;
SameSite=strict
< Cache-Control: max-age=0, no-cache, no-store, must-revalidate
< Expires: 0
< Pragma: no-cache
< Content-Security-Policy: default-src 'self' 'unsafe-eval'
 'unsafe-inline';img-src 'self' data:;frame-ancestors
 https://gghub_prim_vip.frankfurt.goldengate.com;child-src
 https://gghub_prim_vip.frankfurt.goldengate.com blob:;
< X-Content-Type-Options: nosniff
< X-XSS-Protection: 1; mode=block
< X-OGG-Proxy-Version: v1
< Strict-Transport-Security: max-age=31536000 ; includeSubDomains
<
* Connection #0 to host gghub_prim_vip.frankfurt.goldengate.com left intact
{"$schema":"api:standardResponse","links":[{"rel":"canonical",
"href":"https://gghub_prim_vip.frankfurt.goldengate.com/services/v2/config/
health",
"mediaType":"application/json"},{"rel":"self",
"href":"https://gghub_prim_vip.frankfurt.goldengate.com/services/v2/config/
health",
"mediaType":"application/json"},{"rel":"describedby",
"href":"https://gghub_prim_vip.frankfurt.goldengate.com/services/
ServiceManager/v2/metadata-catalog/health",
"mediaType":"application/schema+json"}],"messages":[],
"response":{"$schema":"ogg:health","deploymentName":"ServiceManager",
"serviceName":"ServiceManager","started":"2023-07-27T15:39:41.867Z","healthy":
true,
"criticalResources":
```

```
[{"deploymentName":"gghubl1","name":"adminsrvr","type":"service",
"status":"running","healthy":true},
{"deploymentName":"gghub1","name":"distsrvr",
"type":"service","status":"running","healthy":true},
{"deploymentName":"gghub1",
"name":"recvsrvr","type":"service","status":"running","healthy":true}]}}
*** Success

[root@gghub_prim1 ~]# rm access.cfg
```

> **Note:**
>
> If the environment is using self-signed SSL certificates, add the flag --insecure to the curl command to avoid the error "NSS error -8172 (SEC_ERROR_UNTRUSTED_ISSUER)".

Step 3.6.8 - Remove NGINX default.conf Configuration File

As the `root` OS user on all GGhub GGHUB, remove the default configuration file (`default.conf`) created in `/etc/nginx/conf.d`:

```
[opc@gghub_prim1 ~]$ sudo rm -f /etc/nginx/conf.d/default.conf
[opc@gghub_prim1 ~]$ sudo nginx -s reload
```

Step 3.6.9 - Distribute the GoldenGate NGINX Configuration Files

Once all the reverse proxy configuration files have been created for the GoldenGate Service Managers, they must be copied to the second GoldenGate Hub node.

As the `opc` OS user on the first GGHUB node, distribute the NGINX configuration files to all database nodes:

```
[opc@gghub_prim1 ~]$ sudo tar fczP /tmp/nginx_conf.tar /etc/nginx/conf.d/
 /etc/nginx/ssl/
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ scp /tmp/nginx_conf.tar gghub_prim2:/tmp/.
```

As the `opc` OS user on the second GGHUB node, extract the NGINX configuration files and remove the default configuration file:

```
[opc@gghub_prim2 ~]$ sudo tar fxzP /tmp/nginx_conf.tar
[opc@gghub_prim2 ~]$ sudo rm /etc/nginx/conf.d/default.conf
```

As the `opc` OS user on the second GGHUB node, restart NGINX:

```
[opc@gghub_prim2 ~]$ sudo nginx -t


nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful

[root@gghub_prim2 ~]$ sudo systemctl restart nginx
```

> **Note:**
>
> Repeat all the steps in section 3.6 for the primary and standby GGHUB systems.

Step 3.7 - Securing Oracle GoldenGate Microservices to Restrict Non-Secure Direct Access

After configuring the NGINX reverse proxy with an unsecured Oracle GoldenGate Microservices deployment, the microservices can continue accessing HTTP (non-secure) using the configured microservices port numbers. For example, the following non-secure URL could be used to access the Administration Server: http://*vip-name*:9101.

Oracle GoldenGate Microservices' default behavior for each server (Service Manager, adminserver, pmsrvr. distsrvr, and recsrvr) is to listen using a configured port number on all network interfaces. This is undesirable for more secure installations, where direct access using HTTP to the Microservices needs to be disabled and only permitted using NGINX HTTPS.

Use the following commands to alter the Service Manager and deployment services listener address to use only the localhost address. Access to the Oracle GoldenGate Microservices will only be permitted from the localhost, and any access outside of the localhost will only succeed using the NGINX HTTPS port.

Step 3.7.1 - Stop the Service Manager

As the `grid` OS user on the first GGHUB node, stop the GoldenGate deployment:

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ agctl stop goldengate gghub1
[grid@gghub_prim1 ~]$ agctl status goldengate

Goldengate  instance 'gghub1' is not running
```

Step 3.7.2 - Modify the Service Manager Listener Address

As the `oracle` OS user on the first GGHUB node, modify the listener address with the following commands. Use the correct port number for the Service Manager being altered:

```
[opc@gghub_prim1 ~]$ sudo su - oracle
[oracle@gghub_prim1 ~]$ export OGG_HOME=/u01/app/oracle/goldengate/gg21c
[oracle@gghub_prim1 ~]$ export OGG_VAR_HOME=/mnt/acfs_gg1/deployments/
ggsm01/var
[oracle@gghub_prim1 ~]$ export OGG_ETC_HOME=/mnt/acfs_gg1/deployments/
ggsm01/etc
[oracle@gghub_prim1 ~]$ $OGG_HOME/bin/ServiceManager
 --prop=/config/network/serviceListeningPort
 --value='{"port":9100,"address":"127.0.0.1"}' --type=array --persist --exit
```

Step 3.7.3 - Restart the Service Manager and Deployment

As the `grid` OS user on the first GGHUB node, restart the GoldenGate deployment:

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ agctl start goldengate gghub1
```

```
[grid@gghub_prim1 ~]$ agctl status goldengate

Goldengate instance 'gghub1' is running on exadb-node1
```

Step 3.7.4 - Modify the GoldenGate Microservices listener address

As the `oracle` OS user on the first GGHUB node, modify all the GoldenGate microservices (adminsrvr, pmsrvr, distsrvr, recvsrvr) listening address to localhost for the deployments managed by the Service Manager using the following command:

```
[opc@gghub_prim1 ~]$ sudo chmod g+x /u01/oracle/scripts/secureServices.py
[opc@gghub_prim1 ~]$ sudo su - oracle
[oracle@gghub_prim1 ~]$ /u01/oracle/scripts/secureServices.py http://
localhost:9100
 --user oggadmin

Password for 'oggadmin': <oggadmin_password>

*** Securing deployment - gghub1
Current value of "/network/serviceListeningPort" for "gghub1/adminsrvr" is
9101
Setting new value and restarting service.
New value of "/network/serviceListeningPort" for "gghub1/adminsrvr" is
{
    "address": "127.0.0.1",
    "port": 9101
}.
Current value of "/network/serviceListeningPort" for "gghub1/distsrvr" is 9102
Setting new value and restarting service.
New value of "/network/serviceListeningPort" for "gghub1/distsrvr" is
{
    "address": "127.0.0.1",
    "port": 9102
}.
Current value of "/network/serviceListeningPort" for "gghub1/pmsrvr" is 9104
Setting new value and restarting service.
New value of "/network/serviceListeningPort" for "gghub1/pmsrvr" is
{
    "address": "127.0.0.1",
    "port": 9104
}.
Current value of "/network/serviceListeningPort" for "gghub1/recvsrvr" is 9103
Setting new value and restarting service.
New value of "/network/serviceListeningPort" for "gghub1/recvsrvr" is
{
    "address": "127.0.0.1",
    "port": 9103
}.
```

> **Note:**
>
> To modify a single deployment (adminsrvr, pmsrvr, distsrvr, recvsrvr), add the flag `--deployment` *instance_name*

Step 3.8 - Create a Clusterware Resource to Manage NGINX

Oracle Clusterware needs to have control over starting the NGINX reverse proxy so that it can be started automatically before the GoldenGate deployments are started.

As the `grid` OS user on the first GGHUB node, use the following command to get the application VIP resource name required to create the NGINX resource with a dependency on the underlying network CRS resource:

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ crsctl stat res -w "TYPE = app.appviptypex2.type" |grep
NAME

NAME=gghub_prim_vip
```

As the `root` OS user on the first GGHUB node, use the following command to create a Clusterware resource to manage NGINX. Replace the `HOSTING_MEMBERS` and `CARDINALITY` values to match your environment:

```
[opc@gghub_prim1 ~]$ sudo su -
[root@gghub_prim1 ~]# vi /u01/oracle/scripts/add_nginx.sh

# Run as ROOT
$(grep ^crs_home /etc/oracle/olr.loc | cut -d= -f2)/bin/crsctl add resource
nginx
 -type generic_application
 -attr "ACL='owner:root:rwx,pgrp:root:rwx,other::r--,group:oinstall:r-x,
user:oracle:rwx',EXECUTABLE_NAMES=nginx,START_PROGRAM='/bin/systemctl
 start -f nginx',STOP_PROGRAM='/bin/systemctl stop
 -f nginx',CHECK_PROGRAMS='/bin/systemctl status nginx'
 ,START_DEPENDENCIES='hard(<gghub_prim_vip>)
 pullup(<gghub_prim_vip>)',
STOP_DEPENDENCIES='hard(intermediate:<gghub_prim_vip>)',
 RESTART_ATTEMPTS=0, HOSTING_MEMBERS='<gghub_prim1>,<gghub_prim2>',
CARDINALITY=2"

[root@gghub_prim1 ~]# sh /u01/oracle/scripts/add_nginx.sh
```

The NGINX resource created in this example will run on the named database nodes simultaneously, specified by `HOSTING_MEMBERS`. This is recommended when multiple GoldenGate Service Manager deployments are configured and can independently move between database nodes.

Once the NGINX Clusterware resource is created, the GoldenGate XAG resources need to be altered so that NGINX must be started before the GoldenGate deployments are started.

As the `root` OS user on the first GGHUB node, modify the XAG resources using the following example commands.

```
# Determine the current --file systems parameter:

[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ agctl config goldengate gghub1 |grep -i "file system"
```

```
File System resources needed: acfs_primary

# Modify the --file systems parameter:

[opc@gghub_prim1 ~]$ sudo su -
[root@gghub_prim1 ~]# /u01/app/grid/xag/bin/agctl modify goldengate gghub1
 --filesystems acfs_primary,nginx

[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ agctl config goldengate gghub1 |grep -i "File system"

File System resources needed: acfs_primary,nginx
```

> **Note:**
>
> Repeat the above commands for each XAG GoldenGate registration relying on NGINX.
> Repeat all the steps in section 3.8 for the primary and standby GGHUB systems.

Step 3.9 - Create an Oracle Net TNS Alias for Oracle GoldenGate Database Connections

To provide local database connections for the Oracle GoldenGate processes when switching between nodes, create a TNS alias on **all** nodes of the cluster where Oracle GoldenGate may be started. Create the TNS alias in the `tnsnames.ora` file in the `TNS_ADMIN` directory specified in the deployment creation.

If the source database is a multitenant database, two TNS alias entries are required, one for the container database (CDB) and one for the pluggable database (PDB) that is being replicated. For a target Multitenant database, the TNS alias connects the PDB to where replicated data is being applied. The pluggable database `SERVICE_NAME` should be set to the database service created in an earlier step (refer to Step 2.3: Create the Database Services in Task 2: Prepare a Primary and Standby Base System for GGHub).

As the `oracle` OS user on any database node of the primary and the standby database systems, use dbaascli to find the database domain name and the SCAN name:

```
# Primary DB
[opc@exadb1_node1]$ sudo su - oracle
[oracle@exadb1_node1]$ source db_name.env
[oracle@exadb1_node1]$ dbaascli database getDetails --dbname <db_name>
 |grep 'connectString'

        "connectString" : "<primary_scan_name>:1521/<service_name>"

# Standby DB
[opc@exadb2_node1]$ sudo su - oracle
[oracle@exadb2_node1]$ source db_name.env
[oracle@exadb2_node1]$ dbaascli database getDetails --dbname <db_name>
 |grep 'connectString'
```

```
"connectString" : "<standby_scan_name>:1521/<service_name>"
```

As the `oracle` OS user on all nodes of the primary and standby GGHUB, add the recommended parameters for Oracle GoldenGate in the `sqlnet.ora` file:

```
[opc@gghub_prim1]$ sudo su - oracle
[oracle@gghub_prim1]$ mkdir -p /u01/app/oracle/goldengate/network/admin
[oracle@gghub_prim1]$
cat > /u01/app/oracle/goldengate/network/admin/sqlnet.ora <<EOF
DEFAULT_SDU_SIZE = 2097152
EOF
```

As the `oracle` OS user on all nodes of the primary and standby GGHUB, follow the steps to create the TNS alias definitions:

```
[opc@gghub_prim1 ~]$ sudo su - oracle
[oracle@gghub_prim1 ~]$

cat > /u01/app/oracle/goldengate/network/admin/tnsnames.ora <<EOF

# Source
<source_cbd_service_name>=
     (DESCRIPTION =
         (CONNECT_TIMEOUT=3)(RETRY_COUNT=2)(LOAD_BALANCE=off)(FAILOVER=on)
(RECV_TIMEOUT=30)
         (ADDRESS_LIST =
             (ADDRESS = (PROTOCOL = TCP)(HOST=<primary_scan_name>)
(PORT=1521)))
         (ADDRESS_LIST =
             (ADDRESS = (PROTOCOL = TCP)(HOST=<standby_scan_name>)
(PORT=1521)))
         (CONNECT_DATA=(SERVICE_NAME =
<source_cbd_service_name>.goldengate.com)))

<source_pdb_service_name>=
    (DESCRIPTION =
        (CONNECT_TIMEOUT=3)(RETRY_COUNT=2)(LOAD_BALANCE=off)(FAILOVER=on)
(RECV_TIMEOUT=30)
        (ADDRESS_LIST =
            (ADDRESS = (PROTOCOL = TCP)(HOST=<primary_scan_name>)(PORT=1521)))
        (ADDRESS_LIST =
            (ADDRESS = (PROTOCOL = TCP)(HOST=<standby_scan_name>)(PORT=1521)))
        (CONNECT_DATA=(SERVICE_NAME =
<source_pdb_service_name>.goldengate.com)))

# Target
<target_pdb_service_name>=
    (DESCRIPTION =
        (CONNECT_TIMEOUT=3)(RETRY_COUNT=2)(LOAD_BALANCE=off)(FAILOVER=on)
(RECV_TIMEOUT=30)
        (ADDRESS_LIST =
            (ADDRESS = (PROTOCOL = TCP)(HOST=<primary_scan_name>)(PORT=1521)))
        (ADDRESS_LIST =
            (ADDRESS = (PROTOCOL = TCP)(HOST=<standby_scan_name>)(PORT=1521)))
```

```
        (CONNECT_DATA=(SERVICE_NAME =
<target_pdb_service_name>.goldengate.com)))

EOF

[oracle@gghub_prim1 ~]$ scp /u01/app/oracle/goldengate/network/admin/*.ora
 gghub_prim2:/u01/app/oracle/goldengate/network/admin
```

> **Note:**
>
> When the `tnsnames.ora` or `sqlnet.ora` (located in the `TNS_ADMIN` directory for the Oracle GoldenGate deployment) are modified, the deployment needs to be restarted to pick up the changes.

Task 4: Configure the Oracle GoldenGate Environment

Step 4.1 - Create Database Credentials

With the Oracle GoldenGate deployment created, use the Oracle GoldenGate Administration Service home page to create the database credentials using the above TNS alias names. See figure 4 below for an example of the database credential creation using the TNS alias.

From a client machine with access to the GGHUB, create a ssh tunnel to connect to the Oracle GoldenGate Administration Service:

```
$ ssh -N -L <local_port>:<vip>:443 -p 22 <gghub-node>
```

As the oggadmin user, create the database credentials:

1. Log in into the Administration Service: https://localhost:<localPort>/<instance_name>/adminsrvr.
2. Click **Configuration** under **Administration Service**.
3. Click the plus button to **Add Credentials** under the **Database** tab.
4. Add the required information for the source and target CDB and PDB as shown in the table:

Region	Container	Domain	Alias	User ID
Region 1	CDB	GoldenGate	Reg1_CDB	c##ggadmin@<tns_alias>
Region 1	PDB	GoldenGate	Reg1_PDB	ggadmin@<tns_alias>
Region 2	CDB	GoldenGate	Reg2_CDB	c##ggadmin@<tns_alias>
Region 2	PDB	GoldenGate	Reg2_PDB	ggadmin@<tns_alias>

Step 4.2 - Set Up Schema Supplemental Logging

1. Log in to the Oracle GoldenGate Administration Server.
2. Click **Configuration** under **Administration Service**.
3. Click the **Connect to database** button under **Actions** for the **Source Database (Reg_CDB)**.
4. Click the plus button (Add TRANDATA) to **Add TRANDATA** for the **Schema** or **Tables**.

Step 4.3 - Create the Autostart Profile

Create a new profile to automatically start the Extract and Replicat processes when the Oracle GoldenGate Administration Server is started. Then, restart if any Extract or Replicat processes are abandoned. With GoldenGate Microservices, auto start and restart is managed by Profiles.

Using the Oracle GoldenGate Administration Server GUI, create a new profile that can be assigned to each of the Oracle GoldenGate processes:

1. Log in to the **Administration Service** on the Source and Target GoldenGate.
2. Click on **Profile** under **Administration Service**.
3. Click the **plus (+)** sign next to Profiles on the Managed Process Settings home page.
4. Enter the details as follows:
   - Profile Name: Start_Default
   - Description: Default auto-start/resteart profile
   - Default Profile: Yes
   - Auto Start: Yes
   - Auto Start Options
     - Startup Delay: 1 min
     - Auto Restart: Yes
   - Auto Restart Options
     - Max Retries: 5
     - Retry Delay: 30 sec
     - Retries Window: 30 min
     - Restart on Failure only: Yes
     - Disable Task After Retries Exhausted: Yes
5. Click **Submit**

Step 4.4 - Configure Oracle GoldenGate Processes

When creating Extract, Distribution Paths, and Replicat processes with Oracle GoldenGate Microservices Architecture, all files that need to be shared between the GGHub nodes are already shared with the deployment files stored on a shared file system.

Listed below are essential configuration details recommended for running Oracle GoldenGate Microservices on GGhub for Extract, Distribution Paths, and Replicat processes.

Perform the following sub-steps to complete this step:

- Step 4.4.1 - Extract Configuration
- Step 4.4.2 - Replicat Configuration
- Step 4.4.3 - Distribution Path Configuration
- Step 4.4.4 - Set up a Heartbeat Table for Monitoring Lag Times

The main goal is to prevent data divergence between GoldenGate replicas and their associated standby databases. This section focuses on configuring Extract so that GoldenGate Extract never gets ahead of the standby database which can result in data divergence.

GoldenGate Parameter	Description	Recommendations
`TRANLOGOPTIONS HANDLEDLFAILOVER`	This is mandatory setting for Data Guard configurations that have Oracle GoldenGate to ensure GoldenGate Extract never extract data that has not been received by standby database. The HANDLEDLFAILOVER stands for handle DATA LOSS for Data Guard failover. The following parameter must be added to the Extract process parameter fileto avoid losing transactions and resulting in logical data inconsistencies after data loss Data Guard failover event. When the two primary tried to reconcile, this parameter ensures that all transactions can be reconciled since the new primary (old standby) is not further behind as expected. Prevents Extract from extracting redo data from the source database, and writing to the trail file data that has not yet been applied to the Oracle Data Guard standby database. If this parameter is not specified, after a data loss failover, it is possible to have data in the target database that is not present in the source database, leading to data divergence and logical data inconsistencies.	MANDATORY when the source database is configured with Data Guard in Max Availaibility or Max Performance mode.

GoldenGate Parameter	Description	Recommendations
`TRANLOGOPTIONS FAILOVERTARGETDESTID n`	For multiple standby configurations or cases when Data Guard Fast-Start failover is not enabled, set `FAILOVERTARGETDESTID` to standby demarcated by `LOG_ARCHIV_DEST` to ensure GoldenGate Extract never extract data that has not been received by target standby database. To determine the correct value for `FAILOVERTARGETDESTID`, use the `LOG_ARCHIVE_DEST_N` parameter from the GoldenGate source database which is used for sending redo to the source standby database. For example, if `LOG_ARCHIVE_DEST_2` points to the standby database, then use a value of 2. When not using Data Guard Fast Start Failover (FSFO) in the source database, this parameter Identifies which standby database the Extract process must remain behind, with regard to not extracting redo data that has not yet been applied to the Oracle Data Guard standby database.	MANDATORY when not using FSFO in the source database. To determine the correct value for `FAILOVERTARGETDESTID`, use the `LOG_ARCHIVE_DEST_N` parameter from the GoldenGate source database which is used for sending redo to the source standby database. For example, if `LOG_ARCHIVE_DEST_2` points to the standby database, then use a value of 2.
`TRANLOGOPTIONS HANDLEDLFAILOVER STANDBY_WARNING` *value*	The amount of time before a warning message is written to the Extract report file, if Extract is stalled, due to being unable to query the source database standby apply progress. This can occur after a Data Guard failover when the old primary database is not currently available. The default is 60 seconds.	OPTIONAL if want to adjust the timing of when the warning message is written to the Extract report file. Add `STANDBY_WARNING` *value* to the `TRANLOGOPTIONS HANDLEDLFAILOVER` parameter.
`TRANLOGOPTIONS HANDLEDLFAILOVER STANDBY_ABEND` *value*	The amount of time before Extract abends, if Extract is stalled, due to being unable to query the standby apply progress. The default is 30 minutes.	OPTIONAL if want to adjust the amount of time it takes Extract to abend, when the source database standby is not accessible to enforce the `HANDLEDLFAILOVER` parameter. Add `STANDBY_ABEND` *value* to the `TRANLOGOPTIONS HANDLEDLFAILOVER` parameter.

GoldenGate Parameter	Description	Recommendations
`TRANLOGOPTIONS DLFAILOVER_TIMEOUT value`	The amount of time Extract will run on the new source primary database, after a Data Guard role transition, before it will check the status of the standby database. If standby database is not available after the DLFAILOVER_TIMEOUT, Extract will abend. The default is 300 seconds. NOTE: If during normal operations of the source Oracle Data Guard configuration, the standby database becomes unavailable, Extract will stop extracting data from the source database to prevent possible data divergence with the GoldenGate target database due to the HANDLEDLFAILOVER parameter. The DLFAILOVER_TIMEOUT parameter does not take effect when a Data Guard failover has not occurred, and there are no messages output to the Extract report file.	OPTIONAL. if you want to adjust the amount of time an Extract can run on a new primary source database, after a role transition, when the standby is not yet available to honor the `TRANLOGOPTIONS HANDLEDLFAILOVER` parameter.

Refer to the Reference for Oracle GoldenGate for more information about the Extract `TRANLOGOPTIONS` parameters.

When creating an Extract using the Oracle GoldenGate Administration Service GUI interface, leave the Trail SubDirectory parameter blank so that the trail files are automatically created in the deployment directories stored on the shared file system. The default location for trail files is the /<deployment directory>/var/lib/data directory.

> **Note:**
>
> To capture from a multitenant database, you must use an Extract configured at the root level using a c## account. To apply data into a multitenant database, a separate Replicat is needed for each PDB because a Replicat connects at the PDB level and doesn't have access to objects outside of that PDB.

Step 4.4.1 - Extract Configuration

Create the Extract:

1. Log in to the Oracle GoldenGate Administration Server.
2. Click **Overview** under **Administration Service**.
3. Click the **plus** button to **Add Extract**.
4. Select **Integrated Extract**.
5. Add the required information as follows:
   - Process Name: EXT_1
   - Description: Extract for Region 1 CDB
   - Intent: Unidirection

- Begin: Now
- Trail Name: aa
- Credential Domain: GoldenGate
- Credential Alias: Reg1_CDB
- Register to PDBs: PDB Name

6. Click **Next** and set parameters.

```
EXTRACT ext_1
USERIDALIAS Reg1_CDB DOMAIN GoldenGate
EXTTRAIL aaTRANLOGOPTIONS HANDLEDLFAILOVER
TRANLOGOPTIONS FAILOVERTARGETDESTID 2
SOURCECATALOG PDB_NAME
TABLE OWNER.*;
```

7. Click **Next**.
8. If using CDB Root Capture from PDB, add the `SOURCECATALOG` parameter with the PDB Name.
9. Click **Create and Run**.

> **Note:**
>
> For ADB-D deployments, the extract requires a connection to the PDB rather than the CDB.

See Oracle GoldenGate Extract Failure or Error Conditions Considerations for more information.

Step 4.4.2 - Replicat Configuration

Oracle generally recommends using integrated parallel Replicat which offers better apply performance for most workloads when the GGHub is in the same region as the target Oracle GoldenGate database.

The best apply performance can be achieved when the network latency between the GGHub and the target database is as low as possible. The following configuration is recommended for the remote Replicat running on the Oracle GGHub.

- `APPLY_PARALLELISM` – Disables automatic parallelism, instead of using `MAX_APPLY_PARALLELISM` and `MIN_APPLY_PARALLELISM`, and allows the highest amount of concurrency to the target database. It is recommended to set this as high as possible based on available CPU of the hub and the target database server.
- `MAP_PARALLELISM` – Should be set with a value of 2 to 5. With a larger number of appliers, increasing the Mappers increases the ability to hand work to the appliers.
- `BATCHSQL` – applies DML using array processing which reduces the amount network overheads with a higher latency network. Be aware that if there are many data conflicts, BATCHSQL results in reduced performance, as rollback of the batch operations followed by a re-read from trail file to apply in non-batch mode.

Add a Replicat:

After you've set up your database connections and verified them, you can add a Replicat for the deployment by following these steps:

1. Log in to the Oracle GoldenGate Administration Server.
2. Click theplus (+) sign next to **Replicats** on the Administration Service home page. The Add Replicat page is displayed.
3. Select a Replicat type and click **Next**.
4. Enter the details as follows:
   - Process Name: REP_1
   - Description: Replicat for Region 2 PDB
   - Intent: Unidirectional
   - Credential Domain: GoldenGate
   - Credential Alias: Reg2_PDB
   - Source: Trail
   - Trail Name: aa
   - Begin: Position in Log
   - Checkpoint Table: "GGADMIN"."CHKP_TABLE"
5. Click **Next**.
6. From the **Action Menu**, click **Details** to edit the Replicat **Parameters**:

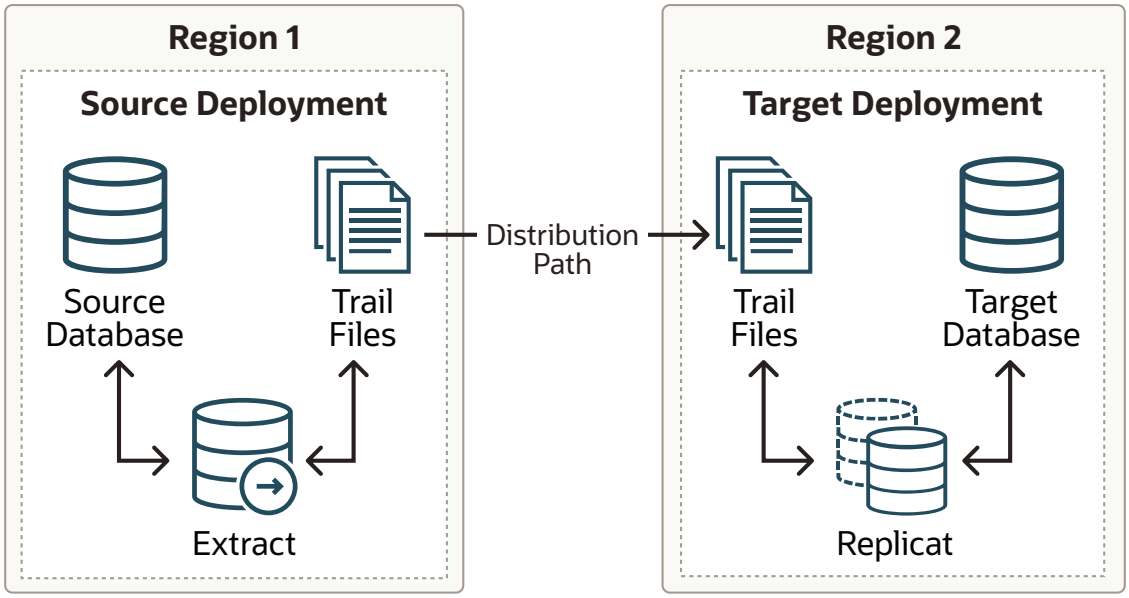
```
REPLICAT REP_1
USERIDALIAS Reg2_PDB DOMAIN GoldenGate
MAP <SOURCE_PDB_NAME>.<OWNER>.*, TARGET <OWNER>.*;
```

7. From the **Action Menu**, click **Start**.

Step 4.4.3 - Distribution Path Configuration

Distribution paths are only necessary when trail files need to be sent to an additional Oracle GoldenGate Hub in a different, or even the same, region as described in the following figure.

Figure 20-4 Oracle GoldenGate Distribution Path



When using Oracle GoldenGate Distribution paths with the NGINX Reverse Proxy, additional steps must be carried out to ensure the path client and server certificates are configured.

More instructions about creating distribution paths are available in *Using Oracle GoldenGate Microservices Architecture*. A step-by-step example is in the following video, "Connect an on-premises Oracle GoldenGate to OCI GoldenGate using NGINX," to correctly configure the certificates.

Here are the steps performed in this sub-step:

- Step 4.4.3.1 - Download the Target Server's Root Certificate, and then upload it to the source Oracle GoldenGate
- Step 4.4.3.2 - Create a user in the Target Deployment for the Source Oracle GoldenGate to use
- Step 4.4.3.3 - Create a Credential in the Source Oracle GoldenGate
- Step 4.4.3.4 - Create a Distribution Path on the Source Oracle GoldenGate to the Target Deployment

Step 4.4.3.1 - Download the Target Server's Root Certificate, and then upload it to the source Oracle GoldenGate

Download the target deployment server's root certificate and add the CA certificate to the source deployment Service Manager.

1. Log in to the **Administration Service** on the Target GoldenGate.
2. Follow "Step 2 - Download the target server's root certificate" in the video "Connect an on-premises Oracle GoldenGate to OCI GoldenGate using NGINX."

Step 4.4.3.2 - Create a user in the Target Deployment for the Source Oracle GoldenGate to use

Create a user in the target deployment for the distribution path to connect to:

1. Log in to the **Administration Service** on the Target GoldenGate.

2. Click on Administrator under Administration Service.
3. Click the plus (+) sign next to Users.
4. Enter the details as follows:
   - Username: ggnet
   - Role: Operator
   - Type: Password
5. Click **Submit**

Step 4.4.3.3 - Create a Credential in the Source Oracle GoldenGate

Create a credential in the source deployment connecting the target deployment with the user created in the previous step. For example, a domain of OP2C and an alias of WSSNET.

1. Log in to the **Administration Service** on the Source Oracle GoldenGate.
2. Click in **Configuration** under **Administration Service.**
3. Click the **plus (+)** sign next to Credentials on the Database home page.
4. Enter the details as follows:
   - Credential Domain: OP2C
   - Credential Alias: wssnet
   - User ID: ggnet
5. Click **Submit**

Step 4.4.3.4 - Create a Distribution Path on the Source Oracle GoldenGate to the Target Deployment

A path is created to send trail files from the Distribution Server to the Receiver Server. You can create a path from the Distribution Service. To add a path for the source deployment:

1. Log in to the **Distribution Service** on the Source Oracle Goldengate.
2. Click the plus (+) sign next to Path on the Distribution Service home page. The Add Path page is displayed.
3. Enter the details as follows:

Option	Description
Path Name	Select a name for the path.
Source: *Trail Name*	Select the Extract name from the drop-down list, which populates the trail name automatically. If it doesn't, enter the trail name you provided while adding the Extract.
Generated Source URI	Specify localhost for the server's name; this allows the distribution path to be started on any of the Oracle RAC nodes.
Target Authentication Method	Use 'UserID Alias'
Target	Set the **Target** transfer protocol to wss (secure web socket). Set the **Target Host** to the target hostname/VIP that will be used for connecting to the target system along with the **Port Number** that NGINX was configured with (default is 443).

Option	Description
Domain	Set the **Domain** to the credential domain created above in Step 11.3.3, for example, OP2C.
Alias	The **Alias** is set to the credential alias wssnet, also created in Step 11.3.3.
Auto Restart Options	Set the distribution path to restart when the Distribution Server starts automatically. This is required, so that manual intervention is not required after a RAC node relocation of the Distribution Server. It is recommended to set the number of **Retries** to 10. Set the **Delay**, which is the time in minutes to pause between restart attempts, to 1.

4. Click **Create Path**.
5. From the Action Menu, click **Start**.

Step 4.4.4 - Set up a Heartbeat Table to Monitor Lag Times

Follow Steps to add Heartbeat Table in OCI GoldenGate to implement the best practices for creating a heartbeat process that can be used to determine where and when lag are developing between a source and target system.

This document walks you through the step-by-step process of creating the necessary tables and added table mapping statements needed to keep track of processing times between a source and target database. Once the information is added into the data flow, the information is then stored in a target table that can be analyzed to determine when and where the lag is being introduced between the source and target systems.

21

Cloud: Oracle GoldenGate Microservices Architecture on Oracle Exadata Database Service Configuration Best Practices

Use these best practices for configuring Oracle GoldenGate Microservices Architecture to work with Oracle Exadata Database Service on Dedicated Infrastructure (ExaDB-D) or Oracle Exadata Database Service on Cloud@Customer (ExaDB-C@C), and with Oracle Database File System (DBFS) or Oracle ASM Cluster File System (ACFS).

See the following topics:

Overview of Oracle GoldenGate Microservices Architecture Configuration on Oracle Exadata Database Service

The target Oracle Exadata Database Service that hosts Oracle GoldenGate Microservices Architecture can act as the source database, the target database, or in some cases, as both source and target databases for Oracle GoldenGate. These best practices are applicable for configuring Oracle GoldenGate Microservices Architecture with Oracle Exadata Database Service on Dedicated Infrastructure or Cloud@Customer.

Follow this roadmap to configure Oracle GoldenGate on Oracle Exadata Database Service on Dedicated Infrastructure (ExaDB-D) or Oracle Exadata Database Service on Cloud@Customer.

- Task 1 - Before You Begin: To configure Oracle GoldenGate on Oracle Exadata Cloud Infrastructure or Cloud@Customer, you need an ExaDB-D or ExaDB-C@C system, CA certificates, and configure some extra software.
- Task 2 - Configure the Oracle Database for GoldenGate: Use best practices to configure the source and target databases in an Oracle GoldenGate replicated environment.
- Task 3 - Create a Shared File System to Store the Oracle GoldenGate Deployment: Set up either Oracle DBFS or Oracle ACFS for configuring HA on Oracle Cloud Infrastructure with Oracle GoldenGate. If your architecture has a GoldenGate replica database protected by a cloud physical standby database (Oracle Data Guard), use Oracle DBFS; otherwise use ACFS.
- Task 4 - Install Oracle GoldenGate: Use best practices to install and configure Oracle GoldenGate components on Oracle Cloud Infrastructure.
- Task 5 - Create the Oracle GoldenGate Deployment: Create a response file to create the GoldenGate deployment using the Oracle GoldenGate Configuration Assistant.
- Task 6 - Configure the Network: Configure virtual cloud network (VCN) components such as private DNS zones, VIP, bastion, security lists and firewalls for Oracle GoldenGate to function properly.
- Task 7 - Configure Oracle Grid Infrastructure Agent: Configure Oracle GoldenGate for HA on Oracle Cloud Infrastructure.
- Task 8 - Configure NGINX Reverse Proxy: Configure reverse proxy and HA by using Nginx.
- Task 9 - Create Oracle Net TNS Alias for Oracle GoldenGate Database Connections: Create a TNS alias to simplify database connectivity for the Oracle GoldenGate processes when switching between Oracle RAC nodes.
- Task 10 - Create a New Profile: Create a new profile to automatically start the Extract and Replicat processes when the Oracle GoldenGate Administration Server is started.
- Task 11 - Configure Oracle GoldenGate Processes: Create and configure Oracle GoldenGate Extract, Replicat, and Path processes need for data replication

Task 1 - Before You Begin

Perform the following steps to complete this task:

- Step 1.1 - Set Up the Oracle Cloud Infrastructure DB System
- Step 1.2 - Download the Required Software
- Step 1.3 - Configure Your System to Install Software from Oracle Linux Yum Server
- Step 1.4 - Secure Deployments Requirements (Certificates)

Step 1.1 - Set Up the Oracle Cloud Infrastructure DB System

To get started, you need an Oracle Exadata Database Service on Dedicated Infrastructure or Cloud@Customer for Oracle GoldenGate deployment.

You can deploy Oracle GoldenGate with an existing ExaDB-D/ExaDB-C@C system or launch a new system, according to your business needs.

For instructions on launching and managing an ExaDB-D system, see Oracle Exadata Database Service on Dedicated Infrastructure or for ExaDB-C@C see Oracle Exadata Database Service on Cloud@Customer.

Step 1.2 - Download the Required Software

1. Create the staging directory to download all the required software.

```
[opc@exadb-node1 ~]$ sudo su -
[root@exadb-node1 ~]# mkdir /u02/app_acfs/goldengate
[root@exadb-node1 ~]# chown oracle:oinstall /u02/app_acfs/goldengate
[root@exadb-node1 ~]# chmod g+w /u02/app_acfs/goldengate
```

2. Download subsequent patches to the base release, go to the Patches and Updates tab of My Oracle Support.
   - See Installing Patches for Oracle GoldenGate Microservices Architecture for more information.
   - The minimum required version is Patch 35214851: Oracle GoldenGate 21.9.0.0.2 Microservices for Oracle
3. Download the latest OPatch release, Patch 6880880, for Oracle Database 21c (21.0.0.0.0) from My Oracle Support Document 2542082.1.
4. Download the Oracle GoldenGate 21c Microservices software, or higher, from Oracle GoldenGate Downloads.
5. Download the Oracle Grid Infrastructure Standalone Agents for Oracle Clusterware 19c, version 10.2 or higher, from Oracle Grid Infrastructure Standalone Agents for Oracle Clusterware.
6. Download the `mount-dbfs-version.zip` file with `mount-dbfs.sh` and `mount-dbfs.conf` from My Oracle Support Document 1054431.1.
7. Download the python script (`secureServices.py`) from My Oracle Support Document 2826001.1.

Step 1.3 - Configure Your System to Install Software from Oracle Linux Yum Server

Oracle Linux yum server hosts software for Oracle Linux and compatible distributions. These instructions help you get started configuring your Linux system for Oracle Linux yum server and installing software via yum.

1. As the `root` OS user, create the file `/etc/yum.repos.d/oracle-public-yum-ol7.repo` with the following contents:

```
[opc@exadb-node1 ~]$ sudo su -
[root@exadb-node1 ~]#
cat > /etc/yum.repos.d/oracle-public-yum-ol7.repo <<EOF
[ol7_latest]
name=Oracle Linux $releasever Latest ($basearch)
baseurl=http://yum$ociregion.oracle.com/repo/OracleLinux/OL7/latest/
\$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
gpgcheck=1
enabled=1
EOF
```

2. As the `root` OS user, follow Doc ID 2397264.1 to modify the configuration file `/etc/yum.conf` and validate the software repositories are enabled:

```
[root@exadb-node1 ~]# yum repolist
repo id             repo name                                   status
```

```
!public_ol7_latest  Oracle Linux 7.9-6.0.1.el7_9 Latest
(x86_64)  19,712+4,957
repolist: 19,992
```

Step 1.4 - Secure Deployments Requirements (Certificates)

A secure deployment involves making RESTful API calls and conveying trail data between the Distribution Server and Receiver Server, over SSL or TLS.

You can use your own existing business certificate from your Certificate Authority (CA) or you might create your own certificates.

Contact your systems administrator to follow your corporate standards to create or obtain the server certificate before proceeding. A separate certificate is required for each VIP and Service Manager pair.

Task 2 - Configure the Oracle Database for GoldenGate

The source and target Oracle GoldenGate databases should be configured using the following recommendations.

Perform the following steps to complete this task:

- Step 2.1 - Database Configuration
- Step 2.2 - Create the Database Replication Administrator User
- Step 2.3 - Create the Database Services

Step 2.1 - Database Configuration

The source and target Oracle GoldenGate databases should be configured using the following recommendations.

1. Enable Oracle GoldenGate replication by setting the database initialization parameter.
2. Source Oracle GoldenGate Database:
   - Run the database in `ARCHIVELOG` mode
   - Enable `FORCE LOGGING` mode
   - Enable minimal supplemental logging
   - Additionally, add schema or table level logging for all replicated objects
3. Configure the streams pool in the System Global Area (SGA) on the source database using the `STREAMS_POOL_SIZE` initialization parameter. The streams pool is only needed on the target database if integrated Replicat will be used.

For the steps on preparing the database for Oracle GoldenGate, refer to Using Oracle GoldenGate Classic Architecture with Oracle Database.

1. As the `oracle` OS user on the source and target systems, issue the following SQL instructions to configure the database:

```
[opc@exadb-node1 ~]$ sudo su - oracle
[oracle@exadb-node1 ~]$ source <db_name>.env
[oracle@exadb-node1 ~]$ sqlplus / as sysdba
SQL> alter system set ENABLE_GOLDENGATE_REPLICATION=true scope=both
sid='*';
SQL> alter system set STREAMS_POOL_SIZE=<SIZE_IN_GB> scope=both sid='*';
```

2. As the `oracle` OS user on the source system, issue the following SQL instructions to configure the database:

```
[opc@exadb-node1 ~]$ sudo su - oracle
[oracle@exadb-node1 ~]$ source <db_name>.env
[oracle@exadb-node1 ~]$ sqlplus / as sysdba
SQL> ALTER DATABASE FORCE LOGGING;
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA;
SQL> ARCHIVE LOG LIST
Database log mode              Archive Mode
Automatic archival             Enabled
Archive destination            USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence     110
Next log sequence to archive   113
Current log sequence           113
```

Step 2.2 - Create the Database Replication Administrator User

The source and target Oracle databases need a GoldenGate Administrator user created, with appropriate privileges assigned:

- For multitenant container database (CDB):
  - Source database, GoldenGate Extract must be configured to connect to a user in the root container database, using a `c##`
  - Target database, a separate GoldenGate administrator user is needed for each pluggable database (PDB). For details about creating a GoldenGate Administrator in an Oracle Multitenant Database, see Configuring Oracle GoldenGate in a Multitenant Container Database.
- For non-CDB databases, see Establishing Oracle GoldenGate Credentials.

1. As the `oracle` OS user on the source system, issue the following SQL instructions to create the database user for Oracle GoldenGate and assign the required privileges:

```
[opc@exadb-node1 ~]$ sudo su - oracle
[oracle@exadb-node1 ~]$ source <db_name>.env
[oracle@exadb-node1 ~]$ sqlplus / as sysdba

# CDB
alter session set container=cdb$root;
create user c##ggadmin identified by "<ggadmin_password>" container=all
default
 tablespace USERS temporary tablespace temp;
alter user c##ggadmin quota unlimited on users;
grant set container to c##ggadmin container=all;
grant alter system to c##ggadmin container=all;
grant create session to c##ggadmin container=all;
grant alter any table to c##ggadmin container=all;
grant resource to c##ggadmin container=all;
exec
dbms_goldengate_auth.grant_admin_privilege('c##ggadmin',container=>'all');

# Source PDB
alter session set container=<PDB_name>;
create user ggadmin identified by "<ggadmin_password>" container=current;
grant create session to ggadmin container=current;
```

```
grant alter any table to ggadmin container=current;
grant resource to ggadmin container=current;
exec dbms_goldengate_auth.grant_admin_privilege('ggadmin');
```

2. As the `oracle` OS user on the target system, issue the following SQL instructions to create the database user for Oracle GoldenGate and assign the required privileges:

```
# Target PDB
[opc@exadb-node1 ~]$ sudo su - oracle
[oracle@exadb-node1 ~]$ source <db_name>.env
[oracle@exadb-node1 ~]$ sqlplus / as sysdba
alter session set container=<PDB_name>;
create user ggadmin identified by "<ggadmin_password>" container=current;
grant alter system to ggadmin container=current;
grant create session to ggadmin container=current;
grant alter any table to ggadmin container=current;
grant resource to ggadmin container=current;
grant dv_goldengate_admin, dv_goldengate_redo_access to ggadmin
container=current;
exec dbms_goldengate_auth.grant_admin_privilege('ggadmin');
```

Step 2.3 - Create the Database Services

A database service is required so that the Oracle Grid Infrastructure Agent will automatically start the Oracle GoldenGate deployment when the database is opened. When DBFS is used for the shared file system, the database service is also used to mount DBFS to the correct RAC instance.

When using a source multitenant database, a separate service is required for the root container database (CDB) and the pluggable database (PDB) that contains the schema being replicated. For a target multitenant database, a single service is required for the PDB.

1. As the `oracle` OS user, create and start the CDB database service using the following command:

```
[oracle@exadb-node1 ~]$ source <db_name>.env
[oracle@exadb-node1 ~]$ srvctl add service -db $ORACLE_UNQNAME
 -service `echo $ORACLE_UNQNAME`_ogg -preferred <SID1> -available <SID2>
 -role PRIMARY
[oracle@exadb-node1 ~]$ srvctl start service -db $ORACLE_UNQNAME
 -service `echo $ORACLE_UNQNAME`_ogg
```

   If your database is part of a multitenant environment, remember to create the service at the pluggable database (PDB).

2. As the `oracle` OS user, create and start the PDB database service using the following command:

```
[oracle@exadb-node1 ~]$ dbaascli database getDetails
 --dbname <db_name> |grep pdbName
      "pdbName" : "<PDB_NAME>",
[oracle@exadb-node1 ~]$ srvctl add service -db $ORACLE_UNQNAME
 -service <PDB_NAME>_ogg -preferred <SID1>,<SID2> -pdb <PDB_NAME> -role
PRIMARY
[oracle@exadb-node1 ~]$ srvctl start service -db $ORACLE_UNQNAME
 -service <PDB_NAME>_ogg
```

3. As the `oracle` OS user, verify that the services are running:

```
[oracle@exadb-node1 ~]$ srvctl status service -d $ORACLE_UNQNAME |grep _ogg
Service <ORACLE_UNQNAME>_ogg is running on instance(s) <SID1>
Service <PDB_NAME>_ogg is running on instance(s) <SID1>
```

See Server Control Utility Reference in *Oracle Real Application Clusters Administration and Deployment Guide* for details about creating database services.

Task 3 - Create a Shared File System to Store the Oracle GoldenGate Deployment

Oracle GoldenGate Microservices Architecture is designed with a simplified installation and deployment directory structure.

- **The installation directory** should be placed on local storage on each database node to minimize downtime during software patching.
- **The deployment directory** which is created during deployment creation using the Oracle GoldenGate Configuration Assistant (oggca.sh), must be placed on a shared file system. The deployment directory contains configuration, security, log, parameter, trail, and checkpoint files.

Placing the deployment in DBFS or Oracle Automatic Storage Management Cluster File System (ACFS) provides the best recoverability and failover capabilities in the event of a system failure. Ensuring the availability of the checkpoint files cluster-wide is essential so that the GoldenGate processes can continue running from their last known position after a failure occurs.

If Oracle GoldenGate will be configured along with Oracle Data Guard, the recommended file system is DBFS. DBFS is contained in the database protected by Data Guard and can be fully integrated with XAG. In the event of a Data Guard role transition, the file system can be automatically mounted on the new primary server, followed by the automated start-up Oracle GoldenGate. This is currently not possible with ACFS since it is not part of the Oracle Data Guard configuration.

> **Note:**
>
> This document does not include steps to configure Oracle GoldenGate with Oracle Data Guard.

If Oracle Data Guard is not present, the recommended file system is ACFS. ACFS is a multi-platform, scalable file system and storage management technology that extends Oracle Automatic Storage Management (Oracle ASM) functionality to support customer files maintained outside the Oracle Database.

Perform one of the following steps to complete this task, based on your file system requirements:

- Step 3a - Oracle Database File System (DBFS)
- Step 3b - Oracle ASM Cluster File System (ACFS)

Step 3a - Oracle Database File System (DBFS)

You must create the DBFS tablespace inside the same database to which the Oracle GoldenGate processes are connected. For example, if an Oracle GoldenGate integrated Extract process is extracted from a database called GGDB, the DBFS tablespace would be located in the same GGDB database.

Create a file system for storing the Oracle GoldenGate deployment files. You should allocate enough trail file disk space to permit storage of up to 12 hours of trail files. Doing this will give sufficient space for trail file generation should a problem occur with the target environment that prevents it from receiving new trail files. The amount of space needed for 12 hours can only be determined by testing trail file generation rates with real production data.

Perform the following sub-steps to complete this step:

- Step 3a.1 - Configuring DBFS on Oracle Exadata Database Service
- Step 3a.2 - Create the DBFS Repository
- Step 3a.3 - (Only for CDB) Create an Entry in TNSNAMES
- Step 3a.4 - Download and Edit the mount-dbfs Scripts
- Step 3a.5 - Register the DBFS Resource with Oracle Clusterware
- Step 3a.6 - Start the DBFS Resource

Step 3a.1 - Configuring DBFS on Oracle Exadata Database Service

1. As the `opc` OS user, add the `grid` user to the `fuse` group:

```
[opc@exadb-node1]$ sudo -u grid $(grep ^crs_home /etc/oracle/olr.loc | cut
-d= -f2)/bin/olsnodes > ~/dbs_group
[opc@exadb-node1]$ dcli -g ~/dbs_group -l opc sudo usermod -a -G fuse grid
```

2. As the `opc` OS user, validate that the file /etc/fuse.conf exists and contains the `user_allow_other` option:

```
[opc@exadb-node1]$ cat /etc/fuse.conf
# mount_max = 1000
# user_allow_other
```

3. Skip this step if the option `user_allow_other` is already in the `/etc/fuse.conf` file. Otherwise, run the following commands as the `opc` OS user to add the option:

```
[opc@exadb-node1]$ dcli -g ~/dbs_group -l opc "echo user_allow_other |
sudo tee -a /etc/fuse.conf"
```

4. As the `opc` OS user, create an empty directory that will be used as the mount point for the DBFS file system:

```
[opc@exadb-node1]$ dcli -g ~/dbs_group -l opc sudo mkdir -p /mnt/dbfs
```

5. As the `opc` OS user, change ownership on the mount point directory so the `grid` OS user can access it:

```
[opc@exadb-node1]$ dcli -g ~/dbs_group -l opc sudo chown
oracle:oinstall /mnt/dbfs
```

Step 3a.2 - Create the DBFS Repository

Create the DBFS repository inside the target database. To create the repository, create a new tablespace within the target PDB to hold the DBFS objects and a database user that will own the objects.

> **Note:**
>
> When using an Oracle Multitenant Database, the DBFS tablespace MUST be created in a Pluggable Database (PDB). It is recommended that you use the same PDB that the GoldenGate Extract or Replicat processes connect to, allowing DBFS to use the same database service created above for its database dependency.

1. As the `oracle` OS user, create the tablespace in the database:

```
[opc@exadb-node1]$ sudo su - oracle
[oracle@exadb-node1]$ source DB_NAME.env
[oracle@exadb-node1]$ sqlplus / as sysdba
SQL> alter session set container=<pdb_name>;
SQL> create bigfile tablespace dbfstb1 datafile size 32g autoextend on
next 8g
 maxsize 300g NOLOGGING EXTENT MANAGEMENT LOCAL AUTOALLOCATE SEGMENT SPACE
 MANAGEMENT AUTO;
SQL> create user dbfs_user identified by "<dbfs_user_password>"
 default tablespace dbfstb1 quota unlimited on dbfstb1;
SQL> grant connect, create table, create view, create procedure,
 dbfs_role to dbfs_user;
```

2. As the `oracle` OS user, create the database objects that will hold DBFS. This script takes two arguments:
   - `dbfstb1`: tablespace for the DBFS database objects
   - `goldengate`: file system name - this can be any string and will appear as a directory under the mount point

```
 [oracle@exadb-node1]$ sqlplus
dbfs_user/"<dbfs_user_password>"@<db_name>_dbfs
SQL> start $ORACLE_HOME/rdbms/admin/dbfs_create_filesystem dbfstb1
goldengate
```

Step 3a.3 - (Only for CDB) Create an Entry in TNSNAMES

1. As the `oracle` OS user, find the database domain name:

```
[opc@exadb-node1]$ sudo su - oracle
[oracle@exadb-node1]$ source DB_NAME.env
[oracle@exadb-node1]$ sqlplus / as sysdba
SQL> show parameter db_domain

NAME                                 TYPE        VALUE
------------------------------------ -----------
------------------------------
db_domain                            string      <db_domain_name>
```

2. As the `oracle` OS user, add a connect entry in `$TNS_ADMIN/tnsnames.ora` file:

```
[oracle@exadb-node1]$ vi $TNS_ADMIN/tnsnames.ora
dbfs =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = IPC)(KEY=LISTENER))
      (CONNECT_DATA =
        (SERVICE_NAME = <pdb_service_name>.<db_domain_name> )
       )
    )
```

3. As the `oracle` OS user, distribute the `$TNS_ADMIN/tnsnames.ora` file to the rest of the nodes:

```
[oracle@exadb-node1 ~]$ /usr/local/bin/dcli -l oracle -g ~/dbs_group
 -f $TNS_ADMIN/tnsnames.ora -d $TNS_ADMIN/
```

Step 3a.4 - Edit the mount-dbfs Scripts

1. Unzip the zip file and edit the variable settings in the file `mount-dbfs.conf` for your environment.
   Comments in the file will help you to confirm the values for these variables:
   - `DBNAME: echo $ORACLE_UNQNAME`
   - `MOUNT_POINT: /mnt/dbfs/goldengate`
   - `ORACLE_HOME (RDBMS ORACLE_HOME directory): echo $ORACLE_HOME`
   - `GRID_HOME (GRID INFRASTRUCTURE HOME directory): echo $(grep ^crs_home /etc/oracle/olr.loc | cut -d= -f2)`
   - `DBFS_PASSWD` (used only if `WALLET=false`)
   - `DBFS_PWDFILE_BASE` (used only if `WALET=false`)
   - `WALLET` (must be `true` or `false`)
   - `TNS_ADMIN` (used only if `WALLET=true` or `PDB): echo $TNS_ADMIN`
   - `DBFS_LOCAL_TNSALIAS` (used only if `WALLET=true`)
   - `IS_PDB` (set to `true` if using PDB)
   - `PDB` (PDB name, if applicable): `PDB name`
   - `PDB_SERVICE` (the database service created in step 2.3, if applicable): *`PDB_SERVICE_NAME`*
   - `MOUNT_OPTIONS: allow_other,direct_io,failover,nolock`
     - The `failover` option forces all file writes to be committed to the DBFS database in an `IMMEDIATE WAIT` mode. This prevents data from getting lost when it has been written into the `dbfs_client` cache, but not yet written to the database at the time of a database or node failure.
     - The `nolock` mount option is required if you use Oracle Database 18c or later versions because of a change in the DBFS file locking, which can cause issues for GoldenGate processes after an Oracle RAC node failure when a file is currently locked.

2. As the `grid` OS user, unzip the `mount-dbfs-<version>.zip` and edit the configuration file `mount-dbfs.conf`:

```
[opc@exadb-node1]$ sudo su - grid
[grid@exadb-node1]$ cd /u02/app_acfs/goldengate
[grid@exadb-node1]$ unzip mount-dbfs-<version>.zip
[grid@exadb-node1]$ vi mount-dbfs.conf
```

Example of `mount-dbfs.conf`:

```
DBNAME=<DB_UNIQUE_NAME>
MOUNT_POINT=/mnt/dbfs/goldengate
DBFS_USER=dbfs_user
GRID_HOME=$(grep ^crs_home /etc/oracle/olr.loc | cut -d= -f2)
if [ -z "${GRID_HOME}" ]; then
    echo "GRID_HOME is unset or set to the empty string"
fi
ORACLE_HOME=$($GRID_HOME/bin/srvctl config database -d $DBNAME |grep
'Oracle home:' | cut -d: -f2 |sed 's/ //g')
if [ -z "${ORACLE_HOME}" ]; then
    echo "ORACLE_HOME is unset or set to the empty string"
fi
LOGGER_FACILITY=user
MOUNT_OPTIONS=allow_other,direct_io,failover,nolock
PERL_ALARM_TIMEOUT=14
DBFS_PASSWD=<DBFS_USER_PASSWORD>
DBFS_PWDFILE_BASE=/tmp/.dbfs-passwd.txt
WALLET=false
TNS_ADMIN=$ORACLE_HOME/network/admin/<DB_NAME>
IS_PDB=true
PDB=<PDB_NAME>
PDB_SERVICE=<PDB_SERVICE_NAME>
```

3. As the `grid` OS user, modify the `mount-dbfs.sh` script to force unmounting of DBFS when the CRS resource is stopped:

```
[grid@exadb-node1]$ vi /u02/app_acfs/goldengate/mount-dbfs.sh

# Change two occurrences of:
$FUSERMOUNT -u $MOUNT_POINT
# To the following:
$FUSERMOUNT -uz $MOUNT_POINT
```

4. As the `opc` OS user, copy `mount-dbfs.conf` (rename it if desired or needed) to the directory `/etc/oracle` on database nodes and set proper permissions on it:

```
[opc@exadb-node1]$ sudo -u grid $(grep ^crs_home /etc/oracle/olr.loc | cut
 -d= -f2)/bin/olsnodes > ~/dbs_group
[opc@exadb-node1]$ /usr/local/bin/dcli -g ~/dbs_group -l opc -d /tmp
 -f /u02/app_acfs/goldengate/mount-dbfs.conf
[opc@exadb-node1]$ /usr/local/bin/dcli -g ~/dbs_group -l opc sudo
 cp /u02/app_acfs/goldengate/mount-dbfs.conf /etc/oracle
[opc@exadb-node1]$ /usr/local/bin/dcli -g ~/dbs_group -l opc sudo
 chown grid:oinstall /etc/oracle/mount-dbfs.conf
```

```
[opc@exadb-node1]$ /usr/local/bin/dcli -g ~/dbs_group -l opc sudo
 chmod 660 /etc/oracle/mount-dbfs.conf
```

5. As the `opc` OS user, copy `mount-dbfs.sh` (rename it if desired or needed) to the proper directory (`$GI_HOME/crs/script`) on database nodes and set proper permissions on it:

```
[opc@exadb-node1]$ /usr/local/bin/dcli -g ~/dbs_group -l opc sudo
 mkdir $(grep ^crs_home /etc/oracle/olr.loc | cut -d= -f2)/crs/script
[opc@exadb-node1]$ /usr/local/bin/dcli -g ~/dbs_group -l opc sudo chown
 grid:oinstall $(grep ^crs_home /etc/oracle/olr.loc | cut -d= -f2)/crs/
script
[opc@exadb-node1]$ /usr/local/bin/dcli -g ~/dbs_group -l grid
 -d $(grep ^crs_home /etc/oracle/olr.loc | cut -d= -f2)/crs/script
 -f /u02/app_acfs/goldengate/mount-dbfs.sh
[opc@exadb-node1]$ /usr/local/bin/dcli -g ~/dbs_group -l grid chmod 770
 $(grep ^crs_home /etc/oracle/olr.loc | cut -d= -f2)/crs/script/mount-
dbfs.sh
```

Step 3a.5 - Register the DBFS Resource with Oracle Clusterware

When registering the resource with Oracle Clusterware, create it as a `cluster_resource`.

The reason for using `cluster_resource` is so the file system can only be mounted on a single node at one time, preventing mounting of DBFS from concurrent nodes creating the potential of concurrent file writes, and causing file corruption problems.

1. As the `grid` OS user, find the resource name for the database service created in a previous step for the DBFS service dependency:

```
[opc@exadb-node1]$ sudo su - grid
[grid@exadb-node1]$ crsctl stat res |grep <PDB_NAME>
NAME=ora.<DB_UNIQUE_NAME>.<SERVICE_NAME>.svc
```

2. As the `oracle` OS user, register the Clusterware resource by running the following script:

```
[opc@exadb-node1]$ sudo su - oracle
[oracle@exadb-node1]$ vi /u02/app_acfs/goldengate/add-dbfs-resource.sh

##### start script add-dbfs-resource.sh
#!/bin/bash
ACTION_SCRIPT=$(grep ^crs_home /etc/oracle/olr.loc | cut -d= -f2)/crs/
script/mount-dbfs.sh
RESNAME=dbfs_mount
DEPNAME=ora.<DB_UNIQUE_NAME>.<SERVICE_NAME>.svc
ORACLE_HOME=$(grep ^crs_home /etc/oracle/olr.loc | cut -d= -f2)
PATH=$ORACLE_HOME/bin:$PATH
export PATH ORACLE_HOME
crsctl add resource $RESNAME \
  -type cluster_resource \
  -attr "ACTION_SCRIPT=$ACTION_SCRIPT, \
         CHECK_INTERVAL=30,RESTART_ATTEMPTS=10, \
         START_DEPENDENCIES='hard($DEPNAME)pullup($DEPNAME)',\
         STOP_DEPENDENCIES='hard($DEPNAME)',\
         SCRIPT_TIMEOUT=300"
##### end script add-dbfs-resource.sh
```

```
[oracle@exadb-node1]$ sh /u02/app_acfs/goldengate/add-dbfs-resource.sh
```

> **Note:**
>
> After creating the `$RESNAME` resource, to stop the `$DBNAME` database when the `$RESNAME` resource is `ONLINE`, you specify the `force` flag when using `srvctl`.
>
> For example: `srvctl stop database -d fsdb -f`

Step 3a.6 - Start the DBFS Resource

As the `grid` OS user, start the resource:

```
 [opc@exadb-node1]$ sudo su - grid
[grid@exadb-node1]$ crsctl start res dbfs_mount -n `hostname`
CRS-2672: Attempting to start 'dbfs_mount' on 'exadb-node1'
CRS-2676: Start of 'dbfs_mount' on 'exadb-node1' succeeded

[grid@exadb-node1]$ crsctl stat res dbfs_mount -t

--------------------------------------------------------------------------------
--
Name           Target  State        Server                   State
details
--------------------------------------------------------------------------------
--
Cluster Resources
--------------------------------------------------------------------------------
--
dbfs_mount
      1        ONLINE  ONLINE       exadb-node1              STABLE
--------------------------------------------------------------------------------
--
```

> **Note:**
>
> Leave the shared file system mounted. It is required for creating the Oracle GoldenGate deployment in a later step.

Step 3b - Oracle ASM Cluster File System (ACFS)
Oracle ACFS is an alternative to DBFS for the shared Oracle GoldenGate files in an Oracle RAC configuration. Create a single ACFS file system for storing the Oracle deployment files.

It is recommended that you allocate enough trail file disk space to permit the storage of up to 12 hours of trail files. Doing this will give sufficient space for trail file generation should a problem occur with the target environment that prevents it from receiving new trail files. The amount of space needed for 12 hours can only be determined by testing trail file generation rates with real production data.

Perform the following sub-steps to complete this step:

- Step 3b.1 - Create the ASM File System
- Step 3b.2 - Make the File System
- Step 3b.3 - Create the Cluster Ready Services (CRS) Resource
- Step 3b.4 - Verify the Currently Configured ACFS File Systems
- Step 3b.5 - Start and Check the Status of the ACFS Resource
- Step 3b.6- Create GoldenGate ACFS Directory

Step 3b.1 - Create the ASM File System

As the `grid` OS user, use `asmcmd` to create the volume:

```
[opc@exadb-node1 ~]$ sudo su - grid
[grid@exadb-node1 ~]$ asmcmd volcreate -G DATAC1 -s 1200G ACFS_GG
```

> **Note:**
>
> Modify the file system size according to the determined size requirements.

Step 3b.2 - Make the File System

1. As the `grid` OS user, use `asmcmd` to confirm the “Volume Device”:

   ```
   [grid@exadb-node1 ~]$ asmcmd volinfo -G DATAC1 ACFS_GG
   ```

   Following is an example of the ACFS volume device output:

   ```
   Diskgroup Name: DATAC1
            Volume Name: ACFS_GG
            Volume Device: /dev/asm/acfs_gg-151
            State: ENABLED
            Size (MB): 1228800
            Resize Unit (MB): 64
            Redundancy: MIRROR
            Stripe Columns: 8
            Stripe Width (K): 1024
            Usage:
            Mountpath:
   ```

2. As the `grid` OS user, make the file system with the following `mkfs` command:

   ```
   [grid@exadb-node1 ~]$ /sbin/mkfs -t acfs /dev/asm/acfs_gg-151
   ```

Step 3b.3 - Create the Cluster Ready Services (CRS) Resource

1. As the `opc` OS user, create the ACFS mount point:

   ```
   [opc@exadb-node1 ~]$ dcli -l opc -g ~/dbs_group sudo mkdir -p /mnt/acfs_gg
   [opc@exadb-node1 ~]$ dcli -l opc -g ~/dbs_group sudo chown
    oracle:oinstall /mnt/acfs_gg
   ```

2. Create the file system resource as the `root` user.

Because the implementation of distributed file locking on ACFS, unlike DBFS, it is acceptable to mount ACFS on more than one Oracle RAC node at any one time.

3. As the `root` OS user, create the ACFS resource for the new ACFS file system:

```
[opc@exadb-node1 ~]$ sudo su -
[root@exadb-node1 ~]# $(grep ^crs_home /etc/oracle/olr.loc | cut -d= -
f2)/bin/srvctl
 add filesystem -device /dev/asm/acfs_gg-151 -volume ACFS_GG -diskgroup
DATAC1
 -path /mnt/acfs_gg -user oracle
```

Step 3b.4 - Verify the Currently Configured ACFS File Systems

As the `grid` OS user, use the following command to view the file system details:

```
[opc@exadb-node1 ~]$ sudo su - grid
[grid@exadb-node1 ~]$ srvctl config filesystem -volume ACFS_GG -diskgroup
DATAC1

Volume device: /dev/asm/acfs_gg-151
Diskgroup name: datac1
Volume name: acfs_gg
Canonical volume device: /dev/asm/acfs_gg-151
Accelerator volume devices:
Mountpoint path: /mnt/acfs_gg
Mount point owner: oracle
Mount point group: oinstall
Mount permissions: owner:oracle:rwx,pgrp:oinstall:r-x,other::r-x
Mount users: grid
Type: ACFS
Mount options:
Description:
ACFS file system is enabled
ACFS file system is individually enabled on nodes:
ACFS file system is individually disabled on nodes:
```

Step 3b.5 - Start and Check the Status of the ACFS Resource

As the `grid` OS user, use the following command to start and check the file system:

```
[grid@exadb-node1 ~]$ srvctl start filesystem -volume ACFS_GG
 -diskgroup DATAC1 -node `hostname`
[grid@exadb-node1 ~]$ srvctl status filesystem -volume ACFS_GG -diskgroup
DATAC1
```

ACFS file system `/mnt/acfs_gg` is mounted on nodes `exadb-node1`

The CRS resource created is named using the format `ora.diskgroup_name.volume_name.acfs`. Using the above file system example, the CRS resource is called `ora.datac1.acfs_gg.acfs`.

To see all ACFS file system CRS resources that currently exist, use the following command.

```
[grid@exadb-node1 ~]$ crsctl stat res -w "((TYPE = ora.acfs.type) OR (TYPE =
ora.acfs_cluster.type))"
```

```
NAME=ora.datac1.acfs_gg.acfs
TYPE=ora.acfs.type
TARGET=ONLINE                    , OFFLINE
STATE=ONLINE on exadb-node1, OFFLINE
NAME=ora.datac1.acfsvol01.acfs
TYPE=ora.acfs.type
TARGET=ONLINE                    , ONLINE
STATE=ONLINE on exadb-node1, ONLINE on exadb-node2
```

Step 3b.6- Create GoldenGate ACFS Directory

As the `grid` OS user, create the directory for storing the Oracle GoldenGate deployments.

```
[opc@exadb-node1 ~]$ sudo su - oracle
[oracle@exadb-node1 ~]$ mkdir -p /mnt/acfs_gg/deployments
```

Refer to the *Oracle Automatic Storage Management Cluster File System Administrator's Guide* for more information about ACFS.

> **Note:**
>
> Leave the shared file system mounted. It is required for creating the Oracle GoldenGate deployment in a later step.

Task 4 - Install Oracle GoldenGate

Install the Oracle GoldenGate software **locally** on all nodes in the Oracle Exadata Database Service configuration that will be part of the Oracle GoldenGate configuration. Make sure the installation directory is **identical** on all nodes.

Perform the following steps to complete this task:

- Step 4.1 - Unzip the Software and Create the Response File for the Installation
- Step 4.2 - Install Oracle GoldenGate
- Step 4-3 - Patch Oracle GoldenGate

Step 4.1 - Unzip the Software and Create the Response File for the Installation

1. As the `oracle` OS user on the first database node, unzip the software:

   ```
   [opc@exadb-node1 ~]$ sudo su - oracle
   [oracle@exadb-node1 ~]$ unzip
    /u02/app_acfs/goldengate/
   213000_fbo_ggs_Linux_x64_Oracle_services_shiphome.zip
    -d /u02/app_acfs/goldengate
   ```

   The software includes an example response file for Oracle Database release 21c and earlier supported releases. Copy the response file to a shared file system, so the same file can be used to install Oracle GoldenGate on all database nodes, and edit the following parameters:

- INSTALL_OPTION=ora21c
- SOFTWARE_LOCATION=/u02/app/oracle/goldengate/gg21c (recommended location)

2. As the oracle OS user on the first database node, copy and edit the response file for the installation.

```
[oracle@exadb-node1 ~]$ cp
 /u02/app_acfs/goldengate/fbo_ggs_Linux_x64_Oracle_services_shiphome/Disk1/
response/oggcore.rsp
 /u02/app_acfs/goldengate
[oracle@exadb-node1 ~]$ vi /u02/app_acfs/goldengate/oggcore.rsp

# Before edit
INSTALL_OPTION=
SOFTWARE_LOCATION=

# After edit
INSTALL_OPTION=ora21c
SOFTWARE_LOCATION=/u02/app/oracle/goldengate/gg21c
```

Step 4.2 - Install Oracle GoldenGate

As the oracle OS user on all database nodes install Oracle GoldenGate:

```
[oracle@exadb-node1 ~]$ cd
 /u02/app_acfs/goldengate/fbo_ggs_Linux_x64_Oracle_services_shiphome/Disk1/
[oracle@exadb-node1 ~]$ ./runInstaller -silent -nowait
 -responseFile /u02/app_acfs/goldengate/oggcore.rsp

Starting Oracle Universal Installer...

Checking Temp space: must be greater than 120 MB.   Actual 32755 MB    Passed
Checking swap space: must be greater than 150 MB.   Actual 16383 MB    Passed
Preparing to launch Oracle Universal Installer from
 /tmp/OraInstall2022-07-08_02-54-51PM. Please wait ...
You can find the log of this install session at:
 /u01/app/oraInventory/logs/installActions2022-07-08_02-54-51PM.log
Successfully Setup Software.
The installation of Oracle GoldenGate Services was successful.
Please check '/u01/app/oraInventory/logs/
silentInstall2022-07-08_02-54-51PM.log'
 for more details.

[oracle@exadb-node1 ~]$ cat
 /u01/app/oraInventory/logs/silentInstall2022-07-08_02-54-51PM.log
The installation of Oracle GoldenGate Services was successful.

[oracle@exadb-node1 ~]$ ssh exadb-node2
[oracle@exadb-node2 ~]$ cd
 /u02/app_acfs/goldengate/fbo_ggs_Linux_x64_Oracle_services_shiphome/Disk1
[oracle@exadb-node2 ~]$ ./runInstaller -silent -nowait
 -responseFile /u02/app_acfs/goldengate/oggcore.rsp

Starting Oracle Universal Installer...

Checking Temp space: must be greater than 120 MB.   Actual 32755 MB    Passed
```

```
Checking swap space: must be greater than 150 MB.   Actual 16383 MB    Passed
Preparing to launch Oracle Universal Installer from
 /tmp/OraInstall2022-07-08_03-54-51PM. Please wait ...
You can find the log of this install session at:
 /u01/app/oraInventory/logs/installActions2022-07-08_03-54-51PM.log
Successfully Setup Software.
The installation of Oracle GoldenGate Services was successful.
Please check '/u01/app/oraInventory/logs/
silentInstall2022-07-08_03-54-51PM.log'
 for more details.

[oracle@exadb-node1 ~]$ cat
 /u01/app/oraInventory/logs/silentInstall2022-07-08_03-54-51PM.log
The installation of Oracle GoldenGate Services was successful.
```

Patch Oracle Goldengate

As the `oracle` OS user on all database nodes, install the latest OPatch:

```
[oracle@exadb-node1 ~]$ unzip -oq -d /u01/app/oracle/goldengate/gg21c
 /u02/app_acfs/goldengate /p6880880_210000_Linux-x86-64.zip
[oracle@exadb-node1 ~]$ cat >> ~/.bashrc <<EOF
export ORACLE_HOME=/u01/app/oracle/goldengate/gg21c
export PATH=$ORACLE_HOME/OPatch:$PATH
EOF
[oracle@exadb-node1 ~]$. ~/.bashrc
[oracle@exadb-node1 ~]$ opatch lsinventory |grep 'Oracle GoldenGate Services'

Oracle GoldenGate Services
21.1.0.0.0

[oracle@gghub_prim1 Disk1]$ opatch version

OPatch Version: 12.2.0.1.37
```

As the `oracle` OS user, run OPatch `prereq` to validate any conflict before applying the patch:

```
[oracle@exadb-node1 ~]$ unzip -oq -d /u02/app_acfs/goldengate
 /u02/app_acfs/goldengate /p35214851_219000OGGRU_Linux-x86-64.zip
[oracle@exadb-node1 ~]$ cd /u02/app_acfs/goldengate/35214851/
[oracle@exadb-node1 35214851]$ opatch prereq CheckConflictAgainstOHWithDetail
-ph ./

Oracle Interim Patch Installer version 12.2.0.1.26
Copyright (c) 2023, Oracle Corporation.  All rights reserved.

PREREQ session
Oracle Home       : /u01/app/oracle/goldengate/gg21c
Central Inventory : /u01/app/oraInventory
   from           : /u01/app/oracle/goldengate/gg21c/oraInst.loc
OPatch version    : 12.2.0.1.26
OUI version       : 12.2.0.9.0
Log file location : /u01/app/oracle/goldengate/gg21c/cfgtoollogs/opatch/
opatch2023-04-21_13-44-16PM_1.log
```

```
Invoking prereq "checkconflictagainstohwithdetail"

Prereq "checkConflictAgainstOHWithDetail" passed.
```

As the `oracle` OS user on all database nodes, patch Oracle GoldenGate Microservices Architecture using OPatch:

```
[oracle@exadb-node1 ~]$ cd /u02/app_acfs/goldengate/35214851/
[oracle@exadb-node1 35214851]$ opatch apply

Oracle Interim Patch Installer version 12.2.0.1.37
Copyright (c) 2023, Oracle Corporation.  All rights reserved.

Oracle Home       : /u01/app/oracle/goldengate/gg21c
Central Inventory : /u01/app/oraInventory
   from           : /u01/app/oracle/goldengate/gg21c/oraInst.loc
OPatch version    : 12.2.0.1.37
OUI version       : 12.2.0.9.0
Log file location : /u01/app/oracle/goldengate/gg21c/cfgtoollogs/opatch/
opatch2023-04-21_19-40-41PM_1.log
Verifying environment and performing prerequisite checks...
OPatch continues with these patches:   35214851

Do you want to proceed? [y|n]
y
User Responded with: Y
All checks passed.

Please shutdown Oracle instances running out of this ORACLE_HOME on the local
system.
(Oracle Home = '/u01/app/oracle/goldengate/gg21c')

Is the local system ready for patching? [y|n]
y
User Responded with: Y
Backing up files...
Applying interim patch '35214851' to OH '/u01/app/oracle/goldengate/gg21c'

Patching component oracle.oggcore.services.ora21c, 21.1.0.0.0...
Patch 35214851 successfully applied.
Log file location: /u01/app/oracle/goldengate/gg21c/cfgtoollogs/opatch/
opatch2023-04-21_19-40-41PM_1.log

OPatch succeeded.

[oracle@exadb-node1 35214851]$ opatch lspatches

35214851;
```

Task 5 - Create the Oracle GoldenGate Deployment

When the Oracle GoldenGate software has been installed, your next step is to create a response file to create the Oracle GoldenGate deployment using the Oracle GoldenGate Configuration Assistant.

Perform the following steps to complete this task:

- Step 5.1 - Create a Response File
- Step 5.2 - Create the GoldenGate Deployment
- Step 5.3 - (only if using DBFS) Move the GoldenGate Deployment Temp Directory

Step 5.1 - Create a Response File

For a silent configuration, as the `oracle` OS user, create and edit the response file `oggca.rsp` to create the Oracle GoldenGate deployment:

```
[opc@exadb-node1 ~]$ sudo su - oracle
[oracle@exadb-node1 ~]$ vi /u02/app_acfs/goldengate/oggca.rsp
oracle.install.responseFileersion=/oracle/install/
rspfmt_oggca_response_schema_v21_1_0

CONFIGURATION_OPTION=ADD
DEPLOYMENT_NAME=<ggNN>
ADMINISTRATOR_USER=oggadmin
ADMINISTRATOR_PASSWORD=<password_for_oggadmin>
SERVICEMANAGER_DEPLOYMENT_HOME=<ACFS or DBFS mount point>/deployments/<ggsmNN>
HOST_SERVICEMANAGER=localhost
PORT_SERVICEMANAGER=9100
SECURITY_ENABLED=false
STRONG_PWD_POLICY_ENABLED=true
CREATE_NEW_SERVICEMANAGER=true
REGISTER_SERVICEMANAGER_AS_A_SERVICE=false
INTEGRATE_SERVICEMANAGER_WITH_XAG=true
EXISTING_SERVICEMANAGER_IS_XAG_ENABLED=false
OGG_SOFTWARE_HOME=/u02/app/oracle/goldengate/gg21c
OGG_DEPLOYMENT_HOME=<ACFS or DBFS mount point>/deployments/<ggNN>
ENV_LD_LIBRARY_PATH=${OGG_HOME}/lib/instantclient:${OGG_HOME}/lib
ENV_TNS_ADMIN=/u02/app/oracle/goldengate/network/admin
FIPS_ENABLED=false
SHARDING_ENABLED=false
ADMINISTRATION_SERVER_ENABLED=true
PORT_ADMINSRVR=9101
DISTRIBUTION_SERVER_ENABLED=true
PORT_DISTSRVR=9102
NON_SECURE_DISTSRVR_CONNECTS_TO_SECURE_RCVRSRVR=false
RECEIVER_SERVER_ENABLED=true
PORT_RCVRSRVR=9103
METRICS_SERVER_ENABLED=true
METRICS_SERVER_IS_CRITICAL=false
PORT_PMSRVR=9104
UDP_PORT_PMSRVR=9105
PMSRVR_DATASTORE_TYPE=BDB
```

```
PMSRVR_DATASTORE_HOME=/u02/app/oracle/goldengate/datastores/<instance_name>
OGG_SCHEMA=<goldengate_database_schema>
```

In the response file, edit the following values appropriately:

- CONFIGURATION_OPTION
- DEPLOYMENT_NAME
- ADMINISTRATOR_USER
- SERVICEMANAGER_DEPLOYMENT_HOME
- OGG_SOFTWARE_HOME
- OGG_DEPLOYMENT_HOME
- ENV_TNS_ADMIN
- OGG_SCHEMA

Step 5.2 - Create the GoldenGate Deployment

As the `oracle` OS user on the first database node, run `oggca.sh` to create the Oracle GoldenGate deployment:

```
[opc@exadb-node1 ~]$ sudo su - oracle
[oracle@exadb-node1 ~]$ export OGG_HOME=/u02/app/oracle/goldengate/gg21c
[oracle@exadb-node1 ~]$ $OGG_HOME/bin/oggca.sh -silent
 -responseFile /u02/app_acfs/goldengate/oggca.rsp
Successfully Setup Software.
```

Step 5.3 - (only if using DBFS) Move the GoldenGate Deployment Temp Directory

After the deployment has been created, if you use DBFS for the shared file system, run the following commands to move the GoldenGate deployment temp directory from DBFS to local storage.

1. As the `oracle` OS user on the first database node, move the GoldenGate deployment temporary directory to the local storage:

   ```
   [opc@exadb-node1 ~]$ sudo su - oracle
   [oracle@exadb-node1 ~]$ dcli -l oracle -g ~/dbs_group mkdir
    -p /u02/app/oracle/goldengate/deployments/<instance_name>
   [oracle@exadb-node1 ~]$ mv
    /mnt/dbfs/goldengate/deployments/<instance_name>/var/temp
    /u02/app/oracle/goldengate/datastores/<instance_name>
   [oracle@exadb-node1 ~]$ ln -s
    /u02/app/oracle/goldengate/deployments/<instance_name>/temp
    /mnt/dbfs/goldengate/deployments/<instance_name>/var/temp
   ```

2. As the `oracle` OS user on the rest of the database nodes, create a directory on the local storage:

   ```
   [oracle@exadb-node2 ~]$ mkdir
    /u02/app/oracle/goldengate/deployments/<instance_name>
   ```

Task 6 - Configure the Network

The way you configure the network depends on your Exadata platform. The first method described in Step 6a applies to ExaDB-D only, and the second method described in Step 6b applies to ExaDB-C@C only.

Perform one of the following steps to complete this task:

- Step 6a - (ExaDB-D only) Configure Oracle Cloud Infrastructure Networking
- Step 6b - (ExaDB-C@C only) Prepare for Application Virtual IP Address Creation

Step 6a - (ExaDB-D only) Configure Oracle Cloud Infrastructure Networking

You must configure virtual cloud network (VCN) components such as private DNS zones, VIP, bastion, security lists, and firewalls for Oracle GoldenGate to function correctly.

To learn more about VCNs and security lists, including instructions for creating them, see Oracle Cloud Infrastructure Networking.

Perform the following sub-steps to complete this step:

- Step 6a.1 - Connect to GoldenGate Microservices Web Interface Using a Private IP
- Step 6a.2 - Create an Application Virtual IP Address (VIP)
- Step 6a.3 - Add Ingress Rule
- Step 6a.4 - Open Port 443 in the Firewall
- Step 6a.5 - Connecting your Source and Target VIP
- Step 6a.5 - Configuring Network Connectivity Between GoldenGate Source and Target
- Step 6a.6 - Configure Private DNS Zones Views and Resolvers

Step 6a.1 - Connect to GoldenGate Microservices Web Interface Using a Private IP
GoldenGate Microservices web interface is only accessible using a private endpoint from within the OCI network or through a bastion host that secures access to OCI resources.

If OCI Bastion service is unavailable in your region, you can use your OCI Compute Instance as a bastion. Follow the steps in OCI Bastion As A Service to create your bastion. You will need one bastion for each region where Oracle GoldenGate Microservices is running.

> **Note:**
>
> After creating a bastion or using a compute instance as a bastion, you need to create an SSH port forwarding session to use `https://localhost:`*`local_port`* to connect to Oracle GoldenGate Microservices.

Step 6a.2 - Create an Application Virtual IP Address (VIP)

A dedicated application VIP is required to allow access to the Oracle GoldenGate Microservices using the same host name, regardless of which Oracle RAC node is hosting the services. An application VIP will also ensure the Oracle GoldenGate Distribution Server can communicate with the Distribution Receiver running the current Oracle RAC node.

The VIP is a cluster resource that Oracle Clusterware manages. The VIP is assigned to a database node and is automatically migrated to another node in the event of a node failure.

Using the Console, assign the VIP to the Oracle Exadata Database Service:

1. Open the navigation menu. Click **Oracle Database**, then click **Exadata on Oracle Public Cloud**.
2. Choose your compartment.
3. Click **Exadata VM Cluster** under Oracle Exadata Database Service on Dedicated Infrastructure.
4. Navigate to the Exadata VM Cluster you want to create the new VIP.
5. Under Resources, click **Virtual IP Address**.
6. Click **Attach Virtual IP Address**.
7. In the Attach Virtual IP Address dialog, enter the following mandatory information:
   - **Subnet**: The client subnet
   - **Virtual IP address hostname**: Use the SCAN DNS Name and replace the SCAN word for Oracle GoldenGate (Example: `exadb-xxxx-ggN`)
8. Click **Create**.

When the Virtual IP Address creation is complete, the status changes from Provisioning to Available, and the assigned IP will be shown in the Virtual IP Address. Make a note of the fully qualified domain name; this is the host name required to connect the source with the target Oracle GoldenGate deployment.

> **Note:**
>
> Adding a new VIP is available in most tenancies; log a Service Request if you have any issues.

Step 6a.3 - Add an Ingress Rule

Using the Console, open ingress port 443 to connect the Oracle GoldenGate service using NGINX as a reverse proxy. For more information, see Working with Security Lists.

After you update the security list, it will have an entry with values similar to the following:

- Source Type: CIDR
- Source CIDR: 0.0.0.0/0
- IP Protocol: TCP
- Source Port Range: All
- Destination Port Range: 443
- Allows: TCP traffic for ports: 443 HTTPS
- Description: Oracle GoldenGate 443

Step 6a.4 - Open Port 443 in the Firewall

As the `opc` OS user, validate if the chains are currently figured to accept traffic:

```
[opc@exadb-node1 ~]$ sudo iptables --list |grep policy

Chain INPUT (policy ACCEPT)
```

```
Chain FORWARD (policy ACCEPT)
Chain OUTPUT (policy ACCEPT)
```

If the policy is `ACCEPT`, you can skip this step and proceed with Task 7. Otherwise, contact your network administrator to update the firewall to open port 443 for ingress activity.

Step 6a.5 - Configuring Network Connectivity Between the GoldenGate Source and Target

You can set up your VCN to access the internet if you like. You can also privately connect your VCN to public Oracle Cloud Infrastructure services such as Object Storage, your on-premises network, or another VCN.

To learn more about whether subnets are public or private, including instructions for creating the connection, see Connectivity Choices in the Oracle Cloud Infrastructure Networking documentation.

Step 6a.6 - Configure Private DNS Zones Views and Resolvers

If the source and target Oracle GoldenGate deployments are in different regions, you must create a private DNS view in the source region with a private zone. This is required for the source Oracle GoldenGate Distribution Path to reach the target Oracle GoldenGate deployment VIP host name.

Follow the steps in Configure private DNS zones views and resolvers to create your private DNS view and zone.

As the `opc` OS user on the source system, use the command `nslookup` to resolve the **Fully qualified domain name** (from Step 6.2) of the target Oracle GoldenGate deployment:

```
[opc@exadb-node1 ~]$ nslookup <target_vip_fully_qualified_domain_name>
Server:         <DNS_IP>
Address:        <DNS_IP>#53

Non-authoritative answer:
Name:   <target_vip_fully_qualified_domain_name>
Address: <target_vip_ip>
```

Step 6b - (ExaDB-C@C only) Prepare for Application Virtual IP Address Creation

A dedicated application VIP is required to allow access to the Oracle GoldenGate Microservices using the same host name, regardless of which Oracle RAC node is hosting the services. An application VIP will also ensure that the Oracle GoldenGate Distribution Server can communicate with the Distribution Receiver running the current Oracle RAC node.

The VIP is a cluster resource that Oracle Clusterware manages. The VIP is assigned to a database node and is automatically migrated to another node in the event of a node failure.

Your system administrator must provide the IP address for the new Application VIP. This IP address must be in the same subnet as the system environment as determined above.

The VIP will be created in the next Task when you configure the Oracle Grid Infrastructure Agent.

Task 7 - Configure Oracle Grid Infrastructure Agent

The following procedure shows you how to configure Oracle Clusterware to manage Oracle GoldenGate using the Oracle Grid Infrastructure Standalone Agent (XAG).

Using XAG automates the mounting of the shared file system (DBFS or ACFS) as well as the stopping and starting of the Oracle GoldenGate deployment when relocating between Oracle RAC nodes.

Perform the following steps to complete this task:

- Step 7.1 - Install the Oracle Grid Infrastructure Standalone Agent
- Step 7.2 - Configure Oracle Grid Infrastructure Agent
- Step 7.2 - Start the Oracle GoldenGate Deployment

Step 7.1 - Install the Oracle Grid Infrastructure Standalone Agent

It is recommended that you install the XAG software as a standalone agent outside the Grid Infrastructure `ORACLE_HOME`. This way, you can use the latest XAG release available, and the software can be updated without impact to the Grid Infrastructure.

XAG must be installed in the same directory on all Oracle RAC database nodes in the system where Oracle GoldenGate is installed.

1. As the `grid` OS user on the first database node, unzip the software and run `sagsetup.sh`:

```
[opc@exadb-node1 ~]$ sudo su - grid
[grid@exadb-node1 ~]$ unzip /u02/app_acfs/goldengate/
p31215432_190000_Generic.zip
 -d /u02/app_acfs/goldengate
[grid@exadb-node1 ~]$ /u02/app_acfs/goldengate/xag/xagsetup.sh --install
 --directory /u01/app/grid/xag --all_nodes
Installing Oracle Grid Infrastructure Agents on: exadb-node1
Installing Oracle Grid Infrastructure Agents on: exadb-node2
Updating XAG resources.
Successfully updated XAG resources.
```

2. Add the location of the newly installed XAG software to the `PATH` variable so that the location of `agctl` is known when the `grid` user logs on to the machine.

```
[grid@exadb-node1 ~]$ grep PATH ~/.bashrc
PATH=
/u01/app/grid/xag/bin:/sbin:/bin:/usr/sbin:/usr/bin:/u01/app/19.0.0.0/grid/
bin:/u01/app/19.0.0.0/grid/OPatch;
 export PATH
```

> **Note:**
>
> It is essential that you ensure that the XAG `bin` directory is specified **before** the Grid Infrastructure `bin` directory to ensure that the correct `agctl` binary is found. This should be set in the `grid` user environment to take effect when logging on, such as in the `.bashrc` file when the Bash shell is in use.

Step 7.2 - Configure Oracle Grid Infrastructure Agent

The following procedure shows you how to configure Oracle Clusterware to manage Oracle GoldenGate using the Oracle Grid Infrastructure Standalone Agent (XAG).

Using XAG automates the mounting of the shared file system (DBFS or ACFS) as well as the stopping and starting of the Oracle GoldenGate deployment when relocating between Oracle RAC nodes.

Oracle GoldenGate must be registered with XAG so that the deployment is started and stopped automatically when the database is started and the file system is mounted.

To register Oracle GoldenGate Microservices Architecture with XAG, use the following command format:

```
agctl add goldengate <instance_name>
--gg_home <GoldenGate_Home>
--service_manager
--config_home <GoldenGate_SvcMgr_Config>
--var_home <GoldenGate_SvcMgr_Var Dir>
--port <port number>
--oracle_home <$OGG_HOME/lib/instantclient>
--adminuser <OGG admin user>
--user <GG instance user>
--group <GG instance group>
--network <network_number>
--ip <ip_address>
--vip_name <vip_name>
--filesystems <CRS_resource_name>
--db_services <service_name>
--use_local_services
--attribute START_TIMEOUT=60
--nodes <node1, node2, ... ,nodeN>
```

Where:

- `--gg_home` specifies the location of the Oracle GoldenGate software.
- `--service_manager` indicates this is a GoldenGate Microservices instance.
- `--config_home` specifies the GoldenGate Service Manager deployment configuration home directory.
- `--var_home` specifies the GoldenGate Service Manager deployment variable home directory.
- `--port` specifies the deployment Service Manager port number.
- `--oracle_home` specifies the location of the Oracle database libraries that are included as part of Oracle GoldenGate 21c and later releases.
  Example: `$OGG_HOME/lib/instantclient`
- `--adminuser` specifies the Oracle GoldenGate Microservices administrator account name.
- `--user` specifies the name of the operating system user that owns the Oracle GoldenGate deployment.
- `--group` specifies the name of the operating system group that owns the Oracle GoldenGate deployment.
- `--network` specifies the network subnet for the VIP.

- `--ip` specifies the IP address for the VIP.
  If you have already created a VIP, specify it using the `--vip_name` *`vip_name`* parameter in place of `--network` and `--ip`.
- `--vip_name` specifies a CRS resource name for an application VIP previously created. This parameter replaces `--network` and `--ip` (optional).
- `--filesystems` specifies the DBFS or ACFS CRS file system resource that must be mounted before the deployment is started.
- `--db_services` specifies the `ora.<database>.<service_name>.svc` service name created in the previous step.
  If you are using Oracle Multitenant Database, specify the PDB database service for Replicat or the CDB database service for an Extract. If using Replicat and Extract, specify both service names, separated by a comma.
- `--use_local_services` specifies that the Oracle GoldenGate instance must be co-located on the same Oracle RAC node where the `db_services` service is running.
- `--attribute` *`name=value`* specifies attributes that can be applied.
  It is recommended that you modify the attribute `START_TIMEOUT=60` to optimize the blackout after a database crash and restart.
- `--nodes` specifies which of the Oracle RAC nodes this GoldenGate instance can run on.
  If Oracle GoldenGate is configured to run on any of the Oracle RAC nodes in the cluster, this parameter should still be used to determine the preferred order of nodes to run Oracle GoldenGate.

Perform one of the following steps to complete this task:

- Step 7.2a - GoldenGate Deployments on DBFS
- Step 7.2b - GoldenGate Deployments on ACFS

Step 7.2a - GoldenGate Deployments on DBFS

1. As the `grid` OS user on the first database node, run the following command to identify the network number:

```
[opc@exadb-node1 ~]$ sudo su - grid
[grid@exadb-node1 ~]$ srvctl config network
Network 1 exists
Subnet IPv4: 10.1.0.0/255.255.255.0/bondeth0, static
Subnet IPv6:
Ping Targets: 10.1.0.1
Network is enabled
Network is individually enabled on nodes:
Network is individually disabled on nodes:
```

2. As the `root` OS user on the first database node, register Oracle GoldenGate Microservices Architecture with XAG using the following command format:

```
[opc@exadb-node1 ~]$ sudo su -

[root@exadb-node1 ~]# /u01/app/grid/xag/bin/agctl add goldengate
<instance_name>     \
--gg_home /u02/app/oracle/goldengate/gg21c
                  \
--service_manager
                   \
```

```
--config_home /mnt/dbfs/deployments/ggsm01/etc/conf
                  \
--var_home /mnt/dbfs/deployments/ggsm01/var
                 \
--port 9100
                  \
--oracle_home /u02/app/oracle/goldengate/gg21c/lib/instantclient
                \
--adminuser oggadmin
                  \
--user oracle
                  \
--group oinstall
                  \
--network 1 --ip <virtual_IP_address>
                  \
--filesystems <dbfs_mount_name>
                  \
--db_services ora.<db_service_name>.svc , ora.<pdb_service_name>.svc
                \
--use_local_services
                  \
--attribute START_TIMEOUT=60
                  \
--nodes <exadb-node1>, <exadb-node2>

Enter password for 'oggadmin' : <oggadmin_password>
```

Step 7.2b - GoldenGate Deployments on ACFS

1. As the `grid` OS user on the first database node, run the following command to identify the network number:

```
[opc@exadb-node1 ~]$ sudo su - grid
[grid@exadb-node1 ~]$ srvctl config network
Network 1 exists
Subnet IPv4: 10.1.0.0/255.255.255.0/bondeth0, static
Subnet IPv6:
Ping Targets: 10.1.0.1
Network is enabled
Network is individually enabled on nodes:
Network is individually disabled on nodes:
```

2. As the `root` OS user on the first database node, register Oracle GoldenGate Microservices Architecture with XAG using the following command format:

```
[root@exadb-node1 ~]# /u01/app/grid/xag/bin/agctl add goldengate
<instance_name>     \
--gg_home /u02/app/oracle/goldengate/gg21c
                  \
--service_manager
                  \
--config_home /mnt/acfs_gg/deployments/ggsm01/etc/conf
                  \
--var_home /mnt/acfs_gg/deployments/ggsm01/var
                  \
```

```
--port 9100
                    \
--oracle_home /u02/app/oracle/goldengate/gg21c/lib/instantclient
                  \
--adminuser oggadmin
                    \
--user oracle
                    \
--group oinstall
                    \
--network 1 --ip <virtual_IP_address>
                   \
--filesystems ora.<acfs_name>.acfs
                     \
--db_services ora.<db_service_name>.svc
                       \
--use_local_services
                    \
--attribute START_TIMEOUT=60
                    \
--nodes <exadb-node1>,<exadb-node2>
```

Step 7.3 - Start the Oracle GoldenGate Deployment

Below are some example `agctl` commands used to manage the Oracle GoldenGate deployment with XAG.

1. As the `grid` OS user, run the following command to start the Oracle GoldenGate deployment:

   ```
   [opc@exadb-node1 ~]$ sudo su - grid
   [grid@exadb-node1 ~]$ agctl start goldengate <instance_name>
   ```

2. As the `grid` OS user, run the following command to check the status of the Oracle GoldenGate:

   ```
   [grid@exadb-node1 ~]$ agctl status goldengate
   Goldengate instance <instance_name> is running on exadb-node1
   ```

3. As the `grid` OS user, run the following command to view the configuration parameters for the Oracle GoldenGate resource:

   ```
   [grid@exadb-node1 ~]$ agctl config goldengate <instance_name>
   Instance name: <instance_name>
   Application GoldenGate location is: /u02/app/oracle/goldengate/gg21c_MS
   Goldengate MicroServices Architecture environment: yes
   Goldengate Service Manager configuration directory:
    /mnt/acfs_gg/deployments/ggsm01/etc/conf
   Goldengate Service Manager var directory: /mnt/acfs_gg/deployments/
   ggsm01/var
   Service Manager Port: 9100
   Goldengate Administration User: oggadmin
   Autostart on DataGuard role transition to PRIMARY: no
   Configured to run on Nodes: exadb-node1 exadb-node2
   ORACLE_HOME location is: /u02/app/oracle/goldengate/gg21c/lib/instantclient
   Database Services needed: ora.<db_unique_name>.<service_name>.svc
   ```

```
[use_local_services]
File System resources needed: ora.datac1.acfs_gg.acfs
Network: 1, IP:NN.NN.NN.NN, User:oracle, Group:oinstall
```

See Oracle Grid Infrastructure Standalone Agents for Oracle Clusterware 11g Rel. 2, 12c, 18c and 19c for more information about Oracle Grid Infrastructure Bundled Agent.

Task 8 - Configure NGINX Reverse Proxy

The Oracle GoldenGate reverse proxy feature allows a single point of contact for all of the Oracle GoldenGate Microservices associated with an Oracle GoldenGate deployment.

Without a reverse proxy, the Oracle GoldenGate deployment microservices are contacted using a URL consisting of a host name or IP address and separate port numbers, one for each of the services.

For example, to contact the Service Manager, you could use http://gghub.example.com:9100, then the Administration Server is http://gghub.example.com:9101, the second Service Manager may be accessed using http://gghub.example.com:9110, and so on.

When running Oracle GoldenGate in a High Availability (HA) configuration on Oracle Exadata Database Service with the Grid Infrastructure agent (XAG), there is a limitation preventing more than one deployment from being managed by a GoldenGate Service Manager. Because of this limitation, creating a separate virtual IP address (VIP) for each Service Manager and deployment pair is recommended. This way, the microservices can be accessed directly using the VIP.

With a reverse proxy, port numbers are not required to connect to the microservices because they are replaced with the deployment name and the host name's VIP. For example, to connect to the console with a web browser, use the URLs.

Service	URL
Service Manager	https://localhost:*localPort*
Administration Server	https://localhost:*localPort*/*instance_name*/adminsrvr
Distribution Server	https://localhost:*localPort*/*instance_name*/distsrvr
Performance Metric Server	https://localhost:*localPort*/*instance_name*/pmsrvr
Receiver Server	https://localhost:*localPort*/*instance_name*/recvsrvr

> **Note:**
>
> To connect to Oracle GoldenGate in OCI, you must create a bastion and an SSH port forwarding session (see Step 6.1). After this, you can connect to the Oracle GoldenGate Services using https://locahost:<localPort>.

A reverse proxy is mandatory to ensure easy access to microservices and enhance security and manageability.

Follow the instructions to install and configure NGINX Reverse Proxy with an SSL connection and ensure all external communication is secure.

> **Note:**
>
> When using CA Signed Certificates with NGINX, make sure the NGINX `ssl_certificate` parameter points to a certificate file that contains the certificates in the correct order of CA signed certificate, intermediate certificate, and root certificate.

Perform the following steps to complete this task:

- Step 8.1 - Install NGINX
- Step 8.2 - Configure NGINX Reverse Proxy
- Step 8.3 - Securing GoldenGate Microservices to Restrict Non-secure Direct Access
- Step 8.4 - Create a Clusterware Resource to Manage NGINX

Step 8.1 - Install NGINX Reverse Proxy Server

1. As the `root` OS user on all nodes, set up the YUM repository by creating the file `/etc/yum.repos.d/nginx.repo` with the following contents:

```
[opc@exadb-node1 ~]$ sudo su -
[root@exadb-node1 ~]# cat > /etc/yum.repos.d/nginx.repo <<EOF
[nginx-stable]
name=nginx stable repo
baseurl=http://nginx.org/packages/rhel/7/\$basearch/
gpgcheck=1
enabled=1
gpgkey=https://nginx.org/keys/nginx_signing.key
module_hotfixes=true
EOF
```

2. As the `root` OS user, run the following commands to install, enable, and start NGINX:

```
[root@exadb-node1 ~]# yum install -y python-requests python-urllib3 nginx
[root@exadb-node1 ~]# systemctl enable nginx
```

3. As the `root` OS user, disable the NGINX repository after the software has been installed:

```
[root@exadb-node1 ~]# yum-config-manager --disable nginx-stable
```

Step 8.2 - Configure NGINX Reverse Proxy

A separate reverse proxy configuration is required for each Oracle GoldenGate Home.

When running multiple Service Managers, the following instructions will provide configuration using a separate VIP for each Service Manager. NGINX uses the VIP to determine which Service Manager an HTTPS connection request is routed to.

An SSL certificate is required for clients to authenticate the server they connect to through NGINX. Contact your systems administrator to follow your corporate standards to create or obtain the server certificate before proceeding. A separate certificate is required for each VIP and Service Manager pair.

> **Note:**
>
> The common name in the CA-signed certificate must match the target hostname/VIP used by NGINX.

Perform the following sub-steps to complete this step:

- Step 8.2.1 - Create the NGINX Configuration File
- Step 8.2.2 - Modify NGINX Configuration Files
- Step 8.2.3 - Install Server Certificates for NGINX
- Step 8.2.4 - Install the NGINX Configuration File
- Step 8.2.5 - Test the New NGINX Configuration
- Step 8.2.6 - Reload NGINX and the New Configuration
- Step 8.2.7 - Test GoldenGate Microservices Connectivity
- Step 8.2.8 - Distribute the GoldenGate NGINX Configuration Files

Step 8.2.1 - Create the NGINX Configuration File

You can configure Oracle GoldenGate Microservices Architecture to use a reverse proxy. Oracle GoldenGate Microservices Architecture includes a script called `ReverseProxySettings` that generates a configuration file for only the NGINX reverse proxy server.

The script requires the following parameters:

- The `--user` parameter should mirror the GoldenGate administrator account specified with the initial deployment creation.
- The GoldenGate administrator password will be prompted.
- The reverse proxy port number specified by the `--port` parameter should be the default HTTPS port number (443) unless you are running multiple GoldenGate Service Managers using the same `--host`. In this case, specify an HTTPS port number that does not conflict with previous Service Manager reverse proxy configurations.
  For example, if you are running two Service Managers using the same hostname/VIP, the first reverse proxy configuration is created with `--port 443 --host hostvip01`, and the second is created with `--port 444 --host hostvip01`.

  If you are using separate hostnames/VIPs, the two Service Manager reverse proxy configurations would be created with `--port 443 --host hostvip01` and `--port 443 --host hostvip02`.
- Lastly, the HTTP port number (9100) should match the Service Manager port number specified during the deployment creation.

Repeat this step for each additional GoldenGate Service Manager.

As the `oracle` OS user, use the following command to create the Oracle GoldenGate NGINX configuration file:

```
[opc@exadb-node1 ~]$ sudo su - oracle
[oracle@exadb-node1 ~]$ export OGG_HOME=/u02/app/oracle/goldengate/gg21c
[oracle@exadb-node1 ~]$ export PATH=$PATH:$OGG_HOME/bin
[oracle@exadb-node1 ~]$ cd /u02/app_acfs/goldengate
[oracle@exadb-node1 ~]$ $OGG_HOME/lib/utl/reverseproxy/ReverseProxySettings
 --user oggadmin --port 443 --output ogg_<instance_name>.conf http://
```

```
localhost:9100
 --host <VIP hostname/IP>
Password: <oggadmin_password>
```

Step 8.2.2 - Modify NGINX Configuration Files

When multiple GoldenGate Service Managers are configured to use their IP/VIPs with the same HTTPS 443 port, some small changes are required to the NGINX reverse proxy configuration files generated in the previous step.

With all Service Managers sharing the same port number, they are independently accessed using their VIP/IP specified by the `--host` parameter.

1. As the `oracle` OS user, determine the deployment name managed by this Service Manager. If not already known, the deployment name is listed in the reverse proxy configuration file:

   ```
   [opc@exadb-node1 ~]$ sudo su - oracle
   [oracle@exadb-node1 ~]$ cd /u02/app_acfs/goldengate
   [oracle@exadb-node1 ~]$ grep "Upstream Servers" ogg_<instance_name>.conf
   ## Upstream Servers for Deployment '<instance_name>'
   ```

   In this example, the deployment is called `SOURCE`.

2. As the `oracle` OS user, change all occurrences of `_ServiceManager` by prepending the deployment name before the underscore:

   ```
   $ sed -i 's/_ServiceManager/<instance_name>_ServiceManager/'
           ogg_<instance_name>.conf
   ```

Step 8.2.3 - Install Server Certificates for NGINX

1. As the `root` OS user, copy the server certificates and key files in the `/etc/nginx/ssl` directory, owned by `root` with file permissions 400 (-r--------):

   ```
   [opc@exadb-node1 ~]$ sudo su -
   [root@exadb-node1 ~]# mkdir /etc/nginx/ssl
   [root@exadb-node1 ~]# chmod 400 /etc/nginx/ssl
   ```

2. As the `root` OS user, set the correct filenames for the certificate and key files for each reverse proxy configuration file generated in Step 8.2.1:

   ```
   [oracle@exadb-node1 ~]$ vi /u02/app_acfs/goldengate/
   ogg_<instance_name>.conf

   # Before
       ssl_certificate         /etc/nginx/ogg.pem;
       ssl_certificate_key     /etc/nginx/ogg.pem;

   # After
       ssl_certificate         /etc/nginx/ssl/server.chained.crt;
       ssl_certificate_key     /etc/nginx/ssl/server.key;
   ```

When using CA-signed certificates, the certificate named with the `ssl_certificate` NGINX parameter must include the 1) CA signed, 2) intermediate, and 3) root certificates in a single file. The order is significant; otherwise, NGINX fails to start and displays the error message:

```
(SSL: error:0B080074:x509 certificate routines: X509_check_private_key:key values
mismatch)
```

The root and intermediate certificates can be downloaded from the CA-signed certificate provider.

The SSL certificate single file can be generated using the following example command:

```
[root@exadb-node1 ~]# cat
 CA_signed_cert.crt intermediate.crt root.crt > server.chained.crt
```

The `ssl_certificate_key` file is generated when creating the Certificate Signing Request (CSR), which is required when requesting a CA-signed certificate.

Step 8.2.4 - Install the NGINX Configuration File

As the `root` OS user, copy the deployment configuration file (or files if multiple files were created in Step 8.2.1) to `/etc/nginx/conf.d` directory:

```
[root@exadb-node1 ~]# mv /u02/app_acfs/goldengate/ogg_<instance_name>.conf
 /etc/nginx/conf.d
```

Step 8.2.5 - Test the New NGINX Configuration

As the `root` OS user, validate the NGINX configuration file.

If there are errors in the file, they will be reported with the following command:

```
[root@exadb-node1 ~]# nginx -t

nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginxconf test is successful
```

Step 8.2.6 - Reload NGINX and the New Configuration

As the `root` OS user, restart NGINX to load the new configuration:

```
[root@exadb-node1 ~]# systemctl restart nginx
```

Step 8.2.7 - Test GoldenGate Microservices Connectivity

1. As the `root` OS user, create a curl configuration file (`access.cfg`) that contains the deployment username and password:

   ```
   [root@exadb-node1 ~]# vi access.cfg
   user = "oggadmin:<password>"
   ```

2. As the `root` OS user, query the health of the deployments using the following command:

   ```
   [root@exadb-node1 ~]# curl -svf
    -K access.cfg https://<VIP hostname/IP>:<port#>/services/v2/config/health
    -XGET && echo -e "\n*** Success"
   ```

Sample output:

```
{"$schema":"api:standardResponse","links":
[{"rel":"canonical","href":"https://gg-prmy-vip1/services/v2/config/
health",
"mediaType":"application/json"},
{"rel":"self","href":"https://gg-prmy-vip1/services/v2/config/health",
"mediaType":"application/json"},{"rel":"describedby",
"href":"https://gg-prmy-vip1/services/ServiceManager/v2/metadata-catalog/
health",
"mediaType":"application/schema+json"}],"messages":[],
"response":{"$schema":"ogg:health","deploymentName":"ServiceManager",
"serviceName":"ServiceManager","started":"2021-12-09T23:33:03.425Z","health
y":true,
"criticalResources":
[{"deploymentName":"SOURCE","name":"adminsrvr","type":"service",
"status":"running","healthy":true},
{"deploymentName":"SOURCE","name":"distsrvr",
"type":"service","status":"running","healthy":true},
{"deploymentName":"SOURCE","name":"recvsrvr","type":"service","status":"run
ning",
"healthy":true}]}}
*** Success ***
```

3. As the `root` OS user, remove the `curl` configuration file (`access.cfg`) that contains the deployment username and password:

```
[root@exadb-node1 ~]# rm access.cfg
rm: remove regular file ‘access.cfg’? y
```

Step 8.2.8 - Distribute the GoldenGate NGINX Configuration Files
When all of the reverse proxy configuration files have been created for the GoldenGate Service Managers, they must be copied to all the database nodes.

1. As the `opc` OS user, distribute the NGINX configuration files to all database nodes:

```
[opc@exadb-node1 ~]$ sudo tar fczP nginx_conf.tar
 /etc/nginx/conf.d/ /etc/nginx/ssl/
[opc@exadb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc -d /tmp
 -f nginx_conf.tar
[opc@exadb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc sudo tar
fxzP
 /tmp/nginx_conf.tar
```

2. As the `opc` OS user, test the new NGINX configuration on all nodes the new configuration files were copied to:

```
[opc@exadb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc sudo nginx -
t

exadb-node1: nginx: the configuration file /etc/nginx/nginx.conf syntax is
ok
exadb-node1: nginx: configuration file /etc/nginx/nginx.conf test is
successful
exadb-node2: nginx: the configuration file /etc/nginx/nginx.conf syntax is
ok
```

```
exadb-node2: nginx: configuration file /etc/nginx/nginx.conf test is
successful
```

3. As the `opc` OS user, restart NGINX to load the new configuration on all nodes:

```
[opc@exadb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc sudo
systemctl
 restart nginx
```

Step 8.3 - Securing GoldenGate Microservices to Restrict Non-secure Direct Access

After configuring the NGINX reverse proxy with an unsecured Oracle GoldenGate Microservices deployment, the microservices can continue accessing HTTP (non-secure) using the configured microservices port numbers.

For example, the following non-secure URL could be used to access the Administration Server: `http://<vip-name>:9101`.

Oracle GoldenGate Microservices' default behavior for each server (Service Manager, `adminserver`, `pmsrvr`. `distsrvr`, and `recsrvr`) is to listen using a configured port number on all network interfaces. This is undesirable for more secure installations, where direct access using HTTP to the microservices needs to be disabled and only permitted using NGINX HTTPS.

Use the following commands to alter the Service Manager and deployment services listener address to use only the localhost address. Access to the Oracle GoldenGate Microservices will only be permitted from the localhost, and any access outside of the localhost will only succeed using the NGINX HTTPS port.

Step 8.3.1 - Stop the Service Manager

As the `grid` OS user, stop the service manager:

```
[opc@exadb-node1 ~]$ sudo su - grid
[grid@exadb-node1 ~]$ agctl stop goldengate <instance_name>
[grid@exadb-node1 ~]$ agctl status goldengate
Goldengate  instance '<instance_name>' is not running
```

Step 8.3.2 - Modify the Service Manager Listener Address

As the `oracle` OS user, modify the listener address with the following commands.

Use the correct port number for the Service Manager being altered. The server will fail to start, ignore the error, and proceed with the next step:

```
[opc@exadb-node1 ~]$ sudo su - oracle
[oracle@exadb-node1 ~]$ export OGG_HOME=/u02/app/oracle/goldengate/gg21c
[oracle@exadb-node1 ~]$ export
 OGG_VAR_HOME=<acfs or dbfs mount point>/deployments/ggsm01/var
[oracle@exadb-node1 ~]$ export OGG_ETC_HOME=<acfs or
 dbfs mount point>/deployments/ggsm01/etc
[oracle@exadb-node1 ~]$ $OGG_HOME/bin/ServiceManager
 --prop=/config/network/serviceListeningPort
 --value='{"port":9100,"address":"127.0.0.1"}'
 --type=array --persist --exit
[oracle@exadb-node1 ~]$
```

Step 8.3.3 - Restart the Service Manager and Deployment

As the `grid` OS user, restart the Service Manager and deployment:

```
[opc@exadb-node1 ~]$ sudo su - grid
[grid@exadb-node1 ~]$ agctl start goldengate <instance_name>
[grid@exadb-node1 ~]$ agctl status goldengate
Goldengate  instance '<instance_name>' is running on exadb-node1
```

Step 8.3.4 - Modify the GoldenGate Microservices listener address

As the `oracle` OS user, modify all the GoldenGate microservices (`adminsrvr`, `pmsrvr`, `distsrvr`, `recvsrvr`) listening address to `localhost` for the deployments managed by the Service Manager using the following command:

```
[opc@exadb-node1 ~]$ sudo su - oracle
[oracle@exadb-node1 ~]$ cd /u02/app_acfs/goldengate
[oracle@exadb-node1 ~]$ chmod u+x secureServices.py
[oracle@exadb-node1 ~]$ ./secureServices.py http://localhost:9100 --user
oggadmin
Password for 'oggadmin': <oggadmin_password>

*** Securing deployment - ogg_deployment

Current value of "/network/serviceListeningPort" for "<instance_name>/
adminsrvr" is
{
    "address": "127.0.0.1",
    "port": 9101
}
Current value of "/network/serviceListeningPort" for "<instance_name>/
distsrvr" is
{
    "address": "127.0.0.1",
    "port": 9102
}
Current value of "/network/serviceListeningPort" for "<instance_name>/pmsrvr"
is
{
    "address": "127.0.0.1",
    "port": 9104
}
Current value of "/network/serviceListeningPort" for "<instance_name>/
recvsrvr" is
{
    "address": "127.0.0.1",
    "port": 9103
}
```

> **Note:**
>
> To modify a single deployment (`adminsrvr`, `pmsrvr`, `distsrvr`, `recvsrvr`), add the flag `--deployment` *`instance_name`*

Step 8.3.5 - Remove NGINX default.conf Configuration File

As the `opc` OS user, remove the default configuration file (`default.conf`) created in `/etc/nginx/conf.d`:

```
[opc@exadb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc sudo rm
 -f /etc/nginx/conf.d/default.conf
[opc@exadb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc sudo nginx -s
reload
```

Step 8.4 - Create a Clusterware Resource to Manage NGINX

Oracle Clusterware needs to have control over starting the NGINX reverse proxy so that it can be started automatically before the Oracle GoldenGate deployments are started.

1. As the `grid` OS user, use the following command to get the network CRS resource name required to create the NGINX resource with a dependency on the underlying network CRS resource:

   ```
   [opc@exadb-node1 ~]$ sudo su - grid
   [grid@exadb-node1 ~]$ crsctl stat res -w "TYPE == ora.network.type"|grep
   NAME

   NAME=ora.net1.network
   ```

2. As the `root` OS user, use the following example command to create a Clusterware resource to manage NGINX. Replace the `HOSTING_MEMBERS` and `CARDINALITY` to match your environment:

   ```
   [opc@exadb-node1 ~]$ sudo su -

   [root@exadb-node1 ~]# $(grep ^crs_home /etc/oracle/olr.loc | cut -d= -
   f2)/bin/crsctl
    add resource nginx -type generic_application -attr
    "ACL='owner:root:rwx,pgrp:root:rwx,other::r--,group:oinstall:r-
   x,user:oracle:rwx',
   EXECUTABLE_NAMES=nginx,START_PROGRAM='/bin/systemctl start
    -f nginx',STOP_PROGRAM='/bin/systemctl stop
    -f nginx',CHECK_PROGRAMS='/bin/systemctl status nginx'
    ,START_DEPENDENCIES='hard(ora.net1.network) pullup(ora.net1.network)',
    STOP_DEPENDENCIES='hard(intermediate:ora.net1.network)',
    RESTART_ATTEMPTS=0, HOSTING_MEMBERS='<exadb-node1, exadb-node2>',
   CARDINALITY=2"
   ```

   The NGINX resource created in this example will run on the named database nodes simultaneously, specified by `HOSTING_MEMBERS`. This is recommended when multiple GoldenGate Service Manager deployments are configured and can independently move between database nodes.

   Once the NGINX Clusterware resource is created, the GoldenGate XAG resources need to be altered so that NGINX must be started before the GoldenGate deployments are started.

3. As the `root` OS user, modify the XAG resources using the following example commands.

   ```
   # Determine the current --filesystems parameter:
   [opc@exadb-node1 ~]$ sudo su - grid
   [grid@exadb-node1 ~]$ agctl config goldengate <instance_name> |grep "File
   ```

```
System"
File System resources needed: <file_system_resource_name>

# Modify the --filesystems parameter:
[opc@exadb-node1 ~]$ sudo su -
[root@exadb-node1 ~]# /u01/app/grid/xag/bin/agctl modify goldengate
<instance_name>
 --filesystems <file_system_resource_name>,nginx
```

4. Repeat the above commands for each XAG GoldenGate registration relying on NGINX.

Task 9 - Create Oracle Net TNS Alias for Oracle GoldenGate Database Connections

To provide local database connections for the Oracle GoldenGate processes when switching between Oracle RAC nodes, create a TNS alias on **all** of the Oracle RAC nodes where Oracle GoldenGate may be started.

Create the TNS alias in the `tnsnames.ora` file in the `TNS_ADMIN` directory specified in the deployment creation.

Perform the following steps to complete this task:

- Step 9.1 - Create the TNS Alias Definitions
- Step 9.2 - Create the Database Credentials

Step 9.1 - Create the TNS Alias Definitions
If the source database is a multitenant database, two TNS alias entries are required, one for the container database (CDB) and one for the pluggable database (PDB) that is being replicated.

For a target Multitenant database, the TNS alias connects the PDB to where replicated data is being applied. The pluggable database `SERVICE_NAME` should be set to the database service created in an earlier step (Step 2.3: Create the Database Services).

1. As the `oracle` OS user, find the database domain name:

```
[opc@exadb-node1]$ sudo su - oracle
[oracle@exadb-node1]$ source DB_NAME.env
[oracle@exadb-node1]$ sqlplus / as sysdba
SQL> show parameter db_domain

NAME                                 TYPE        VALUE
------------------------------------ -----------
------------------------------
db_domain                            string      <db_domain_name>
```

2. As the `oracle` OS user on the first database node, follow the steps to create the TNS alias definitions and distribute them to all database nodes:

```
[opc@exadb-node1 ~]$ sudo su - oracle
[oracle@exadb-node1 ~]$ dcli -l oracle -g ~/dbs_group mkdir -p /u02/app/
oracle/goldengate/network/admin
[oracle@exadb-node1 ~]$ vi /u02/app/oracle/goldengate/network/admin/
tnsnames.ora
```

```
OGGSRV_CDB =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL=IPC)(KEY=LISTENER))
      (CONNECT_DATA =
        (SERVICE_NAME = <cdb_service_name>.<db_domain_name>)
       )
    )
OGGSRV_<PDB_NAME> =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL=IPC)(KEY=LISTENER))
      (CONNECT_DATA =

        (SERVICE_NAME = <pdb_service_name>.<db_domain_name>)
       )
    )
[oracle@exadb-node1 ~]$ /usr/local/bin/dcli -l oracle -g ~/dbs_group -
f /u02/app/oracle/goldengate/network/admin/*.ora -d /u02/app/oracle/
goldengate/network/admin
```

> **Note:**
>
> When the `tnsnames.ora` or `sqlnet.ora` located in the TNS_ADMIN directory for the Oracle GoldenGate deployment are modified; the deployment needs to be restarted to pick up the changes.

Step 9.2 - Create the Database Credentials
With the Oracle GoldenGate deployment created, use the Oracle GoldenGate Administration Service home page to create the database credentials using the above TNS alias names.

As the `oggadmin` user, create the database credentials:

1. Log in into the Administration Service: https://localhost:<localPort>/<instance_name>/adminsrvr
2. Click on **Configuration** under Administration Service.
3. Click the plus button to **Add Credentials**.
4. Add the required information.
   If the source database is a Multitenant Database, create database credentials for the CDB and PDB. If the target database is a Multitenant Database, create a single credential for the PDB.

Task 10 - Create a New Profile

Create a new profile to automatically start the Extract and Replicat processes when the Oracle GoldenGate Administration Server is started.

Then, restart if any Extract or Replicat processes are abandoned. With GoldenGate Microservices, auto start and restart is managed by Profiles.

Using the Oracle GoldenGate Administration Server GUI, create a new profile that can be assigned to each of the Oracle GoldenGate processes:

1. Log in to the **Administration Service** on the Source and Target GoldenGate.
2. Click on **Profile** under Administration Service.
3. **Click the** plus (+) **sign next to Profiles on the Managed Process Settings home page. The Add Profile page is displayed.**
4. Enter the details.
5. Click **Submit**.

Task 11 - Configure Oracle GoldenGate Processes

When creating Extract, Distribution Paths, and Replicat processes with Oracle GoldenGate Microservices Architecture, all files that need to be shared between Oracle RAC nodes are already shared with the deployment files stored on a shared file system (DBFS or ACFS).

Listed below are the essential configuration details recommended for running Oracle GoldenGate Microservices on Oracle RAC for Extract, Distribution Paths, and Replicat processes.

Perform the following steps to complete this task:

- Step 11.1 - Extract Configuration
- Step 11.2 - (DBFS only) Place the Temporary Cache Files on the Shared Storage
- Step 11.3 - Distribution Path Configuration
- Step 11.4 - Replicat Configuration

Step 11.1 - Extract Configuration

When creating an Extract using the Oracle GoldenGate Administration Service GUI interface, leave the `Trail SubDirectory` parameter blank so that the trail files are automatically created in the deployment directories stored on the shared file system. The default location for trail files is the `/<deployment directory>/var/lib/data` directory.

> **Note:**
>
> To capture from a multitenant database, you must use an Extract configured at the root level using a `c##` account. To apply data into a multitenant database, a separate Replicat is needed for each PDB because a Replicat connects at the PDB level and doesn't have access to objects outside of that PDB

Create the database credentials:

1. Log in to the Oracle GoldenGate **Administration Server** in the Source Oracle GoldenGate.
2. Click in Overview under Administration Service.
3. Click the plus button to **Add Extract**.
4. Select Integrated Extract.
5. Add the required information.
6. Click **Next**.

7. If using CDB Root Capture from PDB, add the `SOURCATALOG` parameter with the PDB Name.
8. Click **Create**.

Step 11.2 - (DBFS only) Place the Temporary Cache Files on the Shared Storage

If you are using DBFS for shared storage, and the deployment `var/temp` directory was moved to local storage as described in Task 5 - Create the Oracle GoldenGate Deployment, it is recommended that you use the Extract `CACHEMGR` parameter to place the temporary cache files on the shared storage.

1. As the `oracle` OS user, create a new directory under the DBFS deployment mount point.:

```
[opc@exadb-node1 ~]$ sudo su - oracle
[oracle@exadb-node1 ~]$ mkdir
 /mnt/dbfs/goldengate/deployments/<instance_name>/temp_cache
```

2. Set the Extract parameter to the new directory:

```
CACHEMGR CACHEDIRECTORY
 /mnt/dbfs/goldengate/deployments/<instance_name>/temp_cache
```

More instructions about creating an Extract process are available in Using Oracle GoldenGate Classic Architecture with Oracle Database.

Step 11.3 - Distribution Path Configuration

When using Oracle GoldenGate Distribution paths with the NGINX Reverse Proxy, additional steps must be carried out to ensure the path client and server certificates are configured.

More instructions about creating distribution paths are available in Oracle GoldenGate Microservices Documentation. A step-by-step example is in the following video, "Connect an on-premises Oracle GoldenGate to OCI GoldenGate using NGINX," to correctly configure the certificates.

Perform the following sub-steps to complete this step:

- Step 11.3.1 - Download the Target Server's Root Certificate, and then upload it to the source Oracle GoldenGate
- Step 11.3.2 - Create a user in the Target Deployment for the Source Oracle GoldenGate to use
- Step 11.3.3 - Create a Credential in the Source Oracle GoldenGate
- Step 11.3.4 - Create a Distribution Path on the Source Oracle GoldenGate to the Target Deployment
- Step 11.3.5 - Verify the Connection in the Target Deployment Console Receiver Service

Step 11.3.1 - Download the Target Server's Root Certificate, and then upload it to the source Oracle GoldenGate

Download the target deployment server's root certificate and add the CA certificate to the source deployment Service Manager.

1. Log in to the **Administration Service** on the Target GoldenGate.
2. Follow "Step 2 - Download the target server's root certificate" in the video "Connect an on-premises Oracle GoldenGate to OCI GoldenGate using NGINX."

Step 11.3.2 - Create a user in the Target Deployment for the Source Oracle GoldenGate to use

Create a user in the target deployment for the distribution path to connect to:

1. Log in to the **Administration Service** on the Target GoldenGate.
2. Click **Administrator** under Administration Service.
3. Click the plus (+) sign next to **Users**.
4. Enter the details.

Step 11.3.3 - Create a Credential in the Source Oracle GoldenGate

Create a credential in the source deployment connecting the target deployment with the user created in the previous step. For example, a domain of OP2C and an alias of WSSNET.

1. Log in to the **Administration Service** on the Source Oracle GoldenGate.
2. Click **Configuration** under Administration Service.
3. Click the plus (+) sign next to **Credentials** on the Database home page. The Add Credentials page is displayed.
4. Enter the details.

Step 11.3.4 - Create a Distribution Path on the Source Oracle GoldenGate to the Target Deployment

A path is created to send trail files from the Distribution Server to the Receiver Server. You can create a path from the Distribution Service.

To add a path for the source deployment:

1. Log in to the **Distribution Service** on the Source Oracle Goldengate.
2. Click the plus (+) sign next to **Path** on the Distribution Service home page. The Add Path page is displayed.
3. Enter the details as follows:

Option	Description
Path Name	Select a name for the path.
Source: *Trail Name*	Select the Extract name from the drop-down list, which populates the trail name automatically. If it doesn't, enter the trail name you provided while adding the Extract.
Generated Source URI	Specify localhost for the server's name; this allows the distribution path to be started on any of the Oracle RAC nodes.
Target Authentication Method	Use `UserID Alias`
Target	Set the **Target** transfer protocol to wss (secure web socket). Set the **Target Host** to the target hostname/VIP that will be used for connecting to the target system along with the **Port Number** that NGINX was configured with (default is 443).
Domain	Set the **Domain** to the credential domain created above in Step 11.3.3, for example, OP2C.
Alias	The **Alias** is set to the credential alias wssnet, also created in Step 11.3.3.

Option	Description
Auto Restart Options	Set the distribution path to restart when the Distribution Server starts automatically. This is required, so that manual intervention is not required after an Oracle RAC node relocation of the Distribution Server. It is recommended to set the number of **Retries** to 10. Set the **Delay**, which is the time in minutes to pause between restart attempts, to 1.

4. Click **Create Path**.
5. From the Action Menu, click **Start**.
6. Verify that the Distribution Service is running.

Step 11.3.5 - Verify the Connection in the Target Deployment Console Receiver Service

1. Log in to the **Administration Service** on the Target Deployment Console.
2. Click on **Receiver Service**.

Step 11.4 - Replicat Configuration

The Replicat process receives the trail data and applies it to the database.

Perform the following sub-steps to complete this step:

- Step 11.4.1 - Create the Checkpoint Table
- Step 11.4.2 - Add a Replicat

Step 11.4.1 - Create the Checkpoint Table

The checkpoint table is a required component for Oracle GoldenGate Replicat processes. After connecting to the database from the Credentials page of the Administration Service, you can create the checkpoint table.

Create the checkpoint table in the target deployment:

1. Log in to the **Administration Service** on the Target GoldenGate.
2. Click **Configuration** under Administration Service.
3. Click **Database** and Connect to the target database or PDB.
4. Click the plus (+) sign next to **Checkpoint**. The Add Checkpoint page is displayed.
5. Enter the details.

See About Checkpoint Table for more information about the checkpoint table.

Step 11.4.2 - Add a Replicat

After you set up your database connections and verified them, you can add a Replicat for the deployment by following these steps:

1. Log in to the **Administration Service** on the Target GoldenGate.
2. Click the plus (+) sign next to **Replicats** on the Administration Service home page. The Add Replicat page is displayed.
3. Select a Replicat type and click **Next**.
4. Enter the details as follows:

Option	Description
Process Name	The name of the Replicat process
Credential Domain	Credential domain created in Step 9.2. In our example is `GoldenGate`
Credential Alias	Credential alias created in Step 9.2. Our example is `Target_PDB`
Source	Select the source to use Trail.
Trail Name	A two-character trail name.
Checkpoint Table	Set the use of an existing checkpoint table.

5. Click **Create Path**.
6. From the Action Menu, click **Start**.
7. Verify that the Replicat is running.

22

Cloud MAA Platinum: Oracle GoldenGate Microservices Architecture Integrated with Active Data Guard

The combination and integration of Oracle GoldenGate Microservices and Oracle Data Guard enables you to achieve an MAA Platinum service-level configuration that achieves zero or near zero downtime for all planned and unplanned outages.

Overview

With these configuration and operational best practices, Oracle GoldenGate can be configured to work seamlessly with Oracle Data Guard after any zero data loss or data loss role transition.

By using Database File System (DBFS) as the file system for the Oracle GoldenGate Microservices deployment files, Oracle GoldenGate Extract, Distribution Paths, and Replicat processes continue to stay synchronized with the database after a role transition.

Implement these best practices for configuring Oracle GoldenGate Microservices Architecture on Oracle Exadata Database Service on Dedicated Infrastructure (ExaDB-D), or Oracle Exadata Database Service on Cloud@Customer (ExaDB-C@C), to work seamlessly with Oracle Data Guard, using Oracle Real Application Clusters (Oracle RAC), Oracle Clusterware, and Oracle Database File System (DBFS).

These best practices enable Oracle GoldenGate Microservices replication using a database that is protected by a Data Guard standby, to work following an Oracle Data Guard role transition transparently and seamlessly, no matter which Data Guard protection mode is configured (Maximum Performance, Maximum Availability, or Maximum Protection).

There are several key software requirements:

- Oracle Grid Infrastructure 19c or later
  Oracle Grid Infrastructure provides the necessary components needed to manage high availability for any business-critical applications. Using Oracle Clusterware (a component of Oracle Grid Infrastructure) network, database, and Oracle GoldenGate resources can be managed to provide availability in the event of a failure.
- Oracle Grid Infrastructure Agent version 10.2 or later
  The Oracle Grid Infrastructure Agent leverages the Oracle Grid Infrastructure components to provide integration between Oracle GoldenGate and its dependent resources, such as the database, network, and file system. The agent also integrates Oracle GoldenGate with Oracle Data Guard so that Oracle GoldenGate is restarted on the new primary database following a role transition.
- Oracle Database 19c or later
  Refer to My Oracle Support note 2193391.1 for a full list of recommended Oracle Database patches when using Oracle GoldenGate.
- Oracle GoldenGate Microservices version 21c or later
  Oracle GoldenGate 21c introduces unified build support so a single software installation supports capturing and applying replicated data to multiple major Oracle Database versions (11g Release 2 to 21c). This is possible because an Oracle GoldenGate

installation includes the required Oracle Database client libraries without requiring a separate database `ORACLE_HOME` installation.

- Oracle DBFS to protect and replicate critical Oracle GoldenGate files
  The Oracle Database File System (DBFS) is the only MAA-validated and recommended file system for an Oracle Data Guard and Oracle GoldenGate configuration, because it allows the storage of the required Oracle GoldenGate files, such as the checkpoint and trail files, to be located inside the same database that is protected with Oracle Data Guard, ensuring consistency between the Oracle GoldenGate files and the database in a seamless fashion.

Task 1 - Before You Begin

To get started, complete the following prerequisites:

- Procure Oracle Exadata Database Service on Dedicated Infrastructure or Cloud@Customer for the Oracle GoldenGate deployment.

  You can deploy Oracle GoldenGate with an existing ExaDB-D or ExaDB-C@C system or launch a new system, according to your business needs. For instructions on launching and managing an ExaDB-D system, see Oracle Exadata Database Service on Dedicated Infrastructure or for ExaDB-C@C see Oracle Exadata Database Service on Cloud@Customer.
- Have Oracle GoldenGate configured as detailed in Cloud: Oracle GoldenGate Microservices Architecture on Oracle Exadata Database Service Configuration Best Practices .

  DBFS is required for critical Oracle GoldenGate files when integrating with Data Guard.
- The Oracle Data Guard standby database should also be configured and operational before continuing.

  For more information about Oracle Data Guard see Getting Started with Oracle Data Guard.
- A secure deployment involves making RESTful API calls and conveying trail data between the Distribution Server and Receiver Server, over SSL/TLS. You can use your own existing business certificate from your Certificate Authority (CA) or you might create your own certificates.

  Contact your systems administrator to follow your corporate standards to create or obtain the server certificate before proceeding. A separate certificate is required for each VIP and Service Manager pair.

Task 2 - Configure the Oracle Database for GoldenGate

Perform the following steps to complete this task:

- Step 2.1 - Configure the Standby Database for Oracle GoldenGate
- Step 2.2 - Modify the Primary Database Service
- Step 2.3 - Create the Standby Database Service

Step 2.1 - Configure the Standby Database for Oracle GoldenGate

The standby database initialization parameters should match those of the primary database, as specified in Cloud: Oracle GoldenGate Microservices Architecture on Oracle Exadata Database Service Configuration Best Practices .

This includes the following parameters:

- `ENABLE_GOLDENGATE_REPLICATION=TRUE`
- For Oracle GoldenGate source databases, enable `FORCE LOGGING` mode and enable minimal supplemental logging.
- If an Oracle GoldenGate source database or running integrated Replicat (parallel or non-parallel), configure the `STREAMS_POOL_SIZE`.

Step 2.2 - Modify the Primary Database Service

On the primary database server, validate the existing database services that were created as part of the original Oracle GoldenGate on Oracle Exadata Database Service configuration.

By default, the service role is defined as `PRIMARY`, so that the service is only started when the database becomes the Data Guard primary database role after a role transition.

As the `oracle` OS user on the primary system, validate the service role using the following command:

```
[opc@exapri-node1 ~]$ sudo su - oracle
[oracle@exapri-node1 ~]$ source <db_name>.env
[oracle@exapri-node1 ~]$ srvctl config service -db $ORACLE_UNQNAME |
 egrep 'Service name|role|Pluggable database name'
Service name: <CDB_SERVICE_NAME>
Service role: PRIMARY
Pluggable database name:
Service name: <PDB_SERVICE_NAME>
Service role: PRIMARY
Pluggable database name: <PDB_NAME>
```

If the roles is not `PRIMARY`, modify the service using the following command:

```
[oracle@exapri-node1 ~]$ srvctl modify service -db $ORACLE_UNQNAME
 -service <service_name> -role PRIMARY
```

If your database is part of a multitenant environment, remember to modify both the multitenant container database (CDB) and pluggable database (PDB) services.

Step 2.3 - Create the Standby Database Service
On the standby Oracle Exadata Database Service, a database service is required for the standby database so that the Oracle Grid Infrastructure Agent will automatically start the Oracle GoldenGate deployment when the database is opened with the primary role.

When a source database is in a multitenant environment, a separate service is required for the root container database (CDB) and the pluggable database (PDB) that contains the schema being replicated. For a multitenant environment target database, a single service is required for the PDB.

Create the service in the standby database as it was created on the primary database. It is recommended that you use the same service name as was specified on the primary system. The service must be created as a singleton service, using the -preferred option, because the application Virtual IP address (VIP), DBFS, and Oracle GoldenGate will run on the system node where the service is running.

1. As the `oracle` OS user, get the Fully Qualified Domain Name (FQDN):

```
[opc@exadb-node1 ~]$ sudo su - oracle
[oracle@exadb-node1 ~]$ hostname -f
exadb-node1.<FQDN>
```

2. As the oracle OS user on the standby system, create the service using the following command:

```
[opc@exastb-node1 ~]$ sudo su - oracle
[oracle@exastb-node1 ~]$ source <db_name>.env
[oracle@exastb-node1 ~]$ srvctl add service -db $ORACLE_UNQNAME
 -service <CDB_SERVICE_NAME>.<FQDN> -preferred <SID1> -available <SID2>
 -role PRIMARY
[oracle@exastb-node1 ~]$ srvctl add service -db $ORACLE_UNQNAME
 -service <PDB_SERVICE_NAME>.<FQDN> -preferred <SID1> -available <SID2>
 -pdb <PDB name> -role PRIMARY
```

Task 3 - Configure Oracle Database File System

The Database File System (DBFS) is the only recommended solution when configuring Oracle GoldenGate with Oracle Data Guard.

The DBFS user, tablespace, and file system in the database was previously created in the primary database, as detailed in Cloud: Oracle GoldenGate Microservices Architecture on Oracle Exadata Database Service Configuration Best Practices .

Perform the following steps to complete this task:

- Step 3.1 - Configuring DBFS on Oracle Exadata Database Service
- Step 3.2 - (PDB Only) Create an Entry in TNSNAMES
- Step 3.3 - Copy and Edit the mount-dbfs Scripts from the Primary System
- Step 3.4 - Register the DBFS Resource with Oracle Clusterware

Step 3.1 - Configuring DBFS on Oracle Exadata Database Service

1. As the `opc` OS user on the standby system, add the `grid` user to the `fuse` group:

```
[opc@exastb-node1 ~]$ sudo -u grid
 $(grep ^crs_home /etc/oracle/olr.loc | cut -d= -f2)/bin/olsnodes > ~/
dbs_group
[opc@exadb-node1 ~]$ dcli -g ~/dbs_group -l opc sudo usermod -a -G fuse
grid
```

2. As the `opc` OS user on the standby system, validate that the file `/etc/fuse.conf` exists and contains the `user_allow_other` option:

```
[opc@exastb-node1 ~]$ cat /etc/fuse.conf
# mount_max = 1000
user_allow_other
```

3. As the `opc` OS user on the standby system, skip this step if the option `user_allow_other` is already in the `/etc/fuse.conf` file. Otherwise run the following commands to add the option:

```
[opc@exastb-node1 ~]$ dcli -g ~/dbs_group -l opc "echo user_allow_other |
 sudo tee -a /etc/fuse.conf"
```

4. As the `opc` OS user on the standby system, create an empty directory that will be used as the mount point for the DBFS filesystem.

> **Note:**
>
> It is important that the mount point is identical as the one in the primary system, because the physical location of the Oracle GoldenGate deployment is included within the deployment configuration files.

```
[opc@exastb-node1 ~]$ dcli -g ~/dbs_group -l opc sudo mkdir -p /mnt/dbfs
```

5. As the `opc` OS user on the standby system, change ownership on the mount point directory so the `grid` OS user can access it:

```
[opc@exastb-node1 ~]$ dcli -g ~/dbs_group -l opc
 sudo chown oracle:oinstall /mnt/dbfs
```

Step 3.2 - (PDB Only) Create an Entry in TNSNAMES

1. As the `oracle` OS user on the standby system, add a connect entry in `$TNS_ADMIN/tnsnames.ora` file. Use the PDB service name created in Step 2.3:

```
[oracle@exadb-node1 ~]$ vi $TNS_ADMIN/tnsnames.ora
dbfs =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = IPC)(KEY=LISTENER))
      (CONNECT_DATA =
        (SERVICE_NAME = <PDB_SERVICE_NAME> )
       )
    )
```

2. As the `oracle` OS user, distribute the `$TNS_ADMIN/tnsnames.ora` file to the rest of the nodes:

```
[oracle@exadb-node1 ~]$ /usr/local/bin/dcli -l oracle -g ~/dbs_group
 -f $TNS_ADMIN/tnsnames.ora -d $TNS_ADMIN/
```

Step 3.3 - Copy and Edit the mount-dbfs Scripts from the Primary System

1. As the `root` OS user on the primary system, create a zip file with the files `mount-dbfs.conf` and `mount-dbfs.sh`:

```
[opc@exapri-node1 ~]$ sudo su -
[root@exapri-node1 ~]# zip -j /tmp/mount-dbfs.zip
 $(grep ^crs_home /etc/oracle/olr.loc | cut -d= -f2)/crs/script/mount-
dbfs.sh
```

```
 /etc/oracle/mount-dbfs.conf
  adding: mount-dbfs.sh (deflated 67%)
  adding: mount-dbfs.conf (deflated 58%)
```

2. As the `opc` OS user on the standby system, copy the `mount-dbfs.zip` file from the primary system to the standby system:

```
[opc@exastb-node1 ~]$ scp exapri-node1.oracle.com:/tmp/mount-dbfs.zip /tmp
```

3. As the `opc` OS user on the standby system, unzip the `mount-dbfs.zip` file and edit the configuration file `mount-dbfs.conf`:

```
[opc@exastb-node1 ~]$ unzip /tmp/mount-dbfs.zip -d /tmp
Archive:  /tmp/mount-dbfs.zip
  inflating: /tmp/mount-dbfs.sh
  inflating: /tmp/mount-dbfs.conf
[opc@exastb-node1 ~]$ vi /tmp/mount-dbfs.conf
```

   It is recommended that you place them in the same directory as the primary system. You will need to modify the following parameters in the `mount-dbfs.conf` file to match the standby database:

   - `DBNAME`
   - `TNS_ADMIN`
   - `PDB_SERVICE`

4. As the `opc` OS user on the standby system, copy `mount-dbfs.conf` to the directory `/etc/oracle` on database nodes and set proper permissions on it:

```
[opc@exastb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc -d /tmp
 -f /tmp/mount-dbfs.conf
[opc@exastb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc sudo
 cp /tmp/mount-dbfs.conf /etc/oracle
[opc@exastb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc sudo
 chown oracle:oinstall /etc/oracle/mount-dbfs.conf
[opc@exastb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc sudo
 chmod 660 /etc/oracle/mount-dbfs.conf
```

5. As the `opc` OS user on the standby system, copy `mount-dbfs.sh` to the directory `$GI_HOME/crs/script` on database nodes and set proper permissions on it:

```
[opc@exastb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc sudo mkdir
 $(grep ^crs_home /etc/oracle/olr.loc | cut -d= -f2)/crs/script
[opc@exastb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc sudo chown
 grid:oinstall $(grep ^crs_home /etc/oracle/olr.loc | cut -d= -f2)/crs/
script
[opc@exastb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l grid
 -d $(grep ^crs_home /etc/oracle/olr.loc | cut -d= -f2)/crs/script
 -f /tmp/mount-dbfs.sh
[opc@exastb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l grid chmod 770
 $(grep ^crs_home /etc/oracle/olr.loc | cut -d= -f2)/crs/script/mount-
dbfs.sh
```

Step 3.3 - Register the DBFS Resource with Oracle Clusterware

When registering the resource with Oracle Clusterware, be sure to create it as a cluster_resource. The reason for using cluster_resource is so the file system can only by mounted on a single node at one time, preventing mounting of DBFS from concurrent nodes creating the potential of concurrent file writes, causing file corruption problems.

If using Oracle Multitenant, make sure to use the service name for the same PDB that contains the DBFS repository as was created in the primary database.

1. As the `grid` OS user on the standby system, find the resource name for the database service created in a previous step for the DBFS service dependency:

```
[opc@exastb-node1 ~]$ sudo su - grid
[grid@exastb-node1 ~]$ crsctl stat res |grep <PDB_NAME>
NAME=ora.<DB_UNIQUE_NAME>.<PDB_SERVICE_NAME>.svc
```

2. As the oracle OS user on the standby system, register the Clusterware resource by executing the following script:

```
[opc@exadb-node1 ~]$ sudo su - oracle
[oracle@exadb-node1 ~]$ vi add-dbfs-resource.sh
##### start script add-dbfs-resource.sh
#!/bin/bash
ACTION_SCRIPT=$(grep ^crs_home /etc/oracle/olr.loc | cut -d=
 -f2)/crs/script/mount-dbfs.sh
RESNAME=dbfs_mount
DEPNAME=ora.<DB_UNIQUE_NAME>.<PDB_SERVICE_NAME>.svc
ORACLE_HOME=$(grep ^crs_home /etc/oracle/olr.loc | cut -d= -f2)
PATH=$ORACLE_HOME/bin:$PATH
export PATH ORACLE_HOME
crsctl add resource $RESNAME \
  -type cluster_resource \
  -attr "ACTION_SCRIPT=$ACTION_SCRIPT, \
         CHECK_INTERVAL=30,RESTART_ATTEMPTS=10, \
         START_DEPENDENCIES='hard($DEPNAME)pullup($DEPNAME)',\
         STOP_DEPENDENCIES='hard($DEPNAME)',\
         SCRIPT_TIMEOUT=300"
##### end script add-dbfs-resource.sh
[oracle@exadb-node1 ~]$ sh add-dbfs-resource.sh
```

Note:

After creating the `$RESNAME` resource, in order to stop the `$DBNAME` database when the `$RESNAME` resource is `ONLINE`, you will have to specify the force flag when using `srvctl`.

For example: `srvctl stop database -d $ORACLE_UNQNAME -f`

Task 4 - Install Oracle GoldenGate

Install the Oracle GoldenGate software locally on all nodes in the Oracle Exadata Database Service configuration that will be part of the GoldenGate configuration.

> **Note:**
>
> Make sure the installation directory is the identical on all nodes to match the primary system installation directory.

1. As the `opc` OS user on the standby system, copy the `oggcore.rsp` response file from the primary system to the standby system:

```
[opc@standby_node_1 ~]$ scp
 primary_node_1:/u02/app_acfs/goldengate/oggcore.rsp
 /u02/app_acfs/goldengate
```

2. On the standby system, follow “Step 4.2 - Install Oracle GoldenGate” as detailed in Task 4 - Install Oracle GoldenGate.

Task 5 - Create Oracle GoldenGate Deployment Directories

The Oracle GoldenGate Service Manager and deployment were already created on the primary system, but certain directories and symbolic links need to be configured on the standby system nodes. These directories and symbolic links were created on the primary system, as detailed in Cloud: Oracle GoldenGate Microservices Architecture on Oracle Exadata Database Service Configuration Best Practices .

1. As the `oracle` OS user on the primary system, determine the datastore directory:

```
[opc@exapri-node1 ~]$ sudo su - oracle
[oracle@exapri-node1 ~]$ grep RepoDatastorePath /mnt/dbfs/goldengate
 /deployments/<instance_name>/var/log/pmsrvr.log|uniq
"RepoDatastorePath": "",
       "RepoDatastorePath": "/u02/app/oracle/goldengate/datastores/
<instance_name>",
```

2. As the `oracle` OS user on the standby system, create the directory on all database nodes:

```
[oracle@exastb-node1 ~]$ /usr/local/bin/dcli -l oracle -g ~/dbs_group mkdir
 -p /u02/app/oracle/goldengate/datastores/<gg_deployment_name>
```

Create the Oracle GoldenGate deployment temp directory local storage to match the symbolic link created on the primary system.

3. As the `oracle` OS user on the primary system, determine the datastore directory:

```
[oracle@exapri-node1 ~]$ ls -l
 /mnt/dbfs/goldengate/deployments/<instance_name>/var |grep temp
```

```
lrwxrwxrwx  1 oracle oinstall 49 Oct  3 10:20 temp ->
 /u02/app/oracle/goldengate/deployments/<instance_name>/temp
```

4. As the `oracle` OS user on the standby system, create the same directory on the standby database nodes:

```
[oracle@exastb-node1 ~]$ /usr/local/bin/dcli -l oracle -g ~/dbs_group mkdir
 -p /u02/app/oracle/goldengate/deployments/<instance_name>/temp
```

Task 6 - Network Configuration

On the standby system, follow the instructions in Task 6 - Configure the Network from Chapter Cloud: Oracle GoldenGate Microservices Architecture on Oracle Exadata Database Service Configuration Best Practices .

Task 7 - Configure Standby NGINX Reverse Proxy

The Oracle GoldenGate reverse proxy feature allows a single point of contact for all of the Oracle GoldenGate microservices associated with an Oracle GoldenGate deployment.

Without reverse proxy, the GoldenGate deployment microservices are contacted using a URL consisting of a host name or IP address and separate port numbers, one for each of the services. Reverse proxy is mandatory to ensure easy access to microservices and provide enhanced security and manageability.

Perform the following steps to complete this task:

- Step 7.1 - Install NGINX Reverse Proxy
- Step 7.2 - Copy NGINX Configuration Files from the Primary System
- Step 7.3 - Install Server Certificates for NGINX
- Step 7.4 - Test and Reload the New NGINX Configuration
- Step 7.5 - Distribute the GoldenGate NGINX Configuration Files
- Step 7.6 - Create a Clusterware Resource to Manage NGINX

Step 7.1 - Install NGINX Reverse Proxy

On the standby system, follow the instructions in “Step 8.1 - Install NGINX Reverse Proxy Server” of Task 8 - Configure NGINX Reverse Proxy to install NGINX.

Step 7.2 - Copy NGINX Configuration Files from the Primary System

1. As the `opc` user on the standby system, copy the NGINX configuration file from the primary to the standby database system:

```
[opc@standby_node_1 ~]$ scp
primary_node_1.oracle.com:/etc/nginx/conf.d/ogg_<deployment_name>.conf
 /tmp
```

2. As the `root` user on the standby system, copy the NGINX configuration file from the directory `/tmp` to the directory `/etc/nginx/conf.d`:

```
[opc@exastb-node1 ~]$ sudo su -
[root@exastb-node1 ~]# cp /tmp/ogg_<deployment_name>.conf /etc/nginx/conf.d
```

Step 7.3 - Install Server Certificates for NGINX

The standby system will need a different CA signed certificate due to using a different VIP name/address than the primary system. Contact your systems administrator to follow your corporate standards to create or obtain the server certificate before proceeding. A separate certificate is required for each VIP and Service Manager pair.

1. As the `root` user on the standby system, copy the server CA certificates and key files in the `/etc/nginx/ssl` directory, owned by `root` with file permissions 400 (-r--------):

```
[opc@exastb-node1 ~]$ sudo su -
[root@exastb-node1 ~]# mkdir /etc/nginx/ssl
[root@exastb-node1 ~]# chmod 400 /etc/nginx/ssl
```

2. As the `root` user on the standby system, set the correct filenames for the certificate and key file to match the same the filenames in the NGINX configuration file:

```
[root@exastb-node1 ~]# grep
 ssl_certificate /etc/nginx/conf.d/ogg_<deployment_name>.conf
    ssl_certificate         /etc/nginx/ssl/server.chained.crt;
    ssl_certificate_key     /etc/nginx/ssl/server.key;
```

> **Note:**
>
> If you have copied multiple reverse proxy configuration files copied from the primary system, you will need to repeat this process for each file.

When using CA signed certificates, the certificate named with the `ssl_certificate` NGINX parameter must include the 1) CA signed, 2) intermediate and 3) root certificates in a single file. The order is very important, otherwise NGINX fails to start and displays the error message:

```
(SSL: error:0B080074:x509 certificate routines: X509_check_private_key:key
values mismatch).
```

The root and intermediate certificates can be downloaded from the CA signed certificate provider.

The single file can be generated using the following example command:

```
[root@exastb-node1 ~]# cat CA_signed_cert.crt intermediate.crt root.crt >
 /etc/nginx/ssl/server.chained.crt
```

The `ssl_certificate_key` file is the key file generated when creating the Certificate Signing Request (CSR), which is required when requesting a CA signed certificate.

3. As the `root` user on the standby system, change the `server_name` parameter to the correct VIP name in the reverse proxy configuration file copied from the primary system:

```
[root@exastb-node1 ~]# vi /etc/nginx/conf.d/ogg_<deployment_name>.conf
# Before:
server_name             exapri-vip.oracle.com;
# After:
server_name             exastb-vip.oracle.com;
```

Step 7.4 - Test and Reload the New NGINX Configuration

Because the VIP will not be running on the standby system until the primary database service is running, there is a parameter that needs to be set in the `/etc/sysctl.conf` file.

1. As the `opc` user on the standby system, add the following parameter to the file `/etc/sysctl.conf`:

```
[opc@exastb-node1 ~]$ sudo vi /etc/sysctl.conf
# allow processes to bind to the non-local address
net.ipv4.ip_nonlocal_bind = 1
```

2. As the `opc` user on the standby system, distribute the `/etc/sysctl.conf` file:

```
[opc@exastb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc
 -d /tmp -f /etc/sysctl.conf
[opc@exastb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc sudo
 cp /tmp/sysctl.conf /etc/sysctl.conf
```

3. As the `opc` user on the standby system, reload the modified configuration:

```
[opc@exastb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc sudo sysctl
 -p /etc/sysctl.conf
```

4. As the `opc` user on the standby system, validate the NGINX configuration file to detect any errors in the configuration. If there are errors in the file, they will be reported by the following command:

```
[opc@exastb-node1 ~]$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginxconf test is successful
```

5. As the `opc` user on the standby system, restart NGINX with the new configuration:

```
[opc@exastb-node1 ~]$ sudo systemctl restart nginx
```

Step 7.5 - Distribute the GoldenGate NGINX Configuration Files

Once all the reverse proxy configuration files have been created for the GoldenGate Service Managers, they need to be copied to all the database nodes.

1. As the `opc` OS user on the standby system, distribute the NGINX configuration files to all the database nodes:

```
[opc@exastb-node1 ~]$ sudo tar fczP nginx_conf.tar
 /etc/nginx/conf.d/ /etc/nginx/ssl/
[opc@exastb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc
```

```
 -d /tmp -f nginx_conf.tar
[opc@exastb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc
 sudo tar fxzP /tmp/nginx_conf.tar
```

2. As the `opc` OS user on the standby system, test the new NGINX configuration on all nodes the new configuration files were copied to:

```
[opc@exastb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc sudo nginx
-t

exastb-node1: nginx: the configuration file /etc/nginx/nginx.conf syntax
is ok
exastb-node1: nginx: configuration file /etc/nginx/nginx.conf test is
successful
exastb-node2: nginx: the configuration file /etc/nginx/nginx.conf syntax
is ok
exastb-node2: nginx: configuration file /etc/nginx/nginx.conf test is
successful
```

3. As the `opc` OS user on the standby system, restart NGINX to load the new configuration on all nodes:

```
[opc@exastb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc
 sudo systemctl restart nginx
```

Step 7.6 - Create a Clusterware Resource to Manage NGINX

Oracle Clusterware needs to have control over starting the NGINX reverse proxy so that it can be started automatically before the GoldenGate deployments are started.

1. As the `grid` OS user on the standby system, use the following command to get the network CRS resource name required to create the NGINX resource with a dependency on the underlying network CRS resource:

```
[opc@exastb-node1 ~]$ sudo su - grid
[grid@exastb-node1 ~]$ crsctl stat res -w "TYPE == ora.network.type"|grep
NAME
NAME=ora.net1.network
```

2. As the `root` user on the standby system, use the following example command to create a Clusterware resource to manage NGINX. Replace `HOSTING_MEMBERS` and `CARDINALITY` to match your environment:

```
[opc@exastb-node1 ~]$ sudo su -

[root@exastb-node1 ~]# $(grep ^crs_home /etc/oracle/olr.loc | cut -d=
 -f2)/bin/crsctl add resource nginx -type generic_application -attr
 "ACL='owner:root:rwx,pgrp:root:rwx,other::r--,group:oinstall:r-
x,user:oracle:rwx',
EXECUTABLE_NAMES=nginx,START_PROGRAM='/bin/systemctl start -f nginx',
STOP_PROGRAM='/bin/systemctl stop -f nginx',
CHECK_PROGRAMS='/bin/systemctl status nginx' ,
START_DEPENDENCIES='hard(ora.net1.network) pullup(ora.net1.network)',
STOP_DEPENDENCIES='hard(intermediate:ora.net1.network)',
```

```
RESTART_ATTEMPTS=0,
HOSTING_MEMBERS='<exastb-node1, exastb-node2>', CARDINALITY=2"
```

The NGINX resource created in this example will run on the named database nodes at the same time, specified by `HOSTING_MEMBERS`. This is recommended when multiple GoldenGate Service Manager deployments are configured, and they can independently move between database nodes.

Task 8 - Configure Oracle Grid Infrastructure Agent

The following procedure shows you how to configure Oracle Clusterware to manage Oracle GoldenGate using the Oracle Grid Infrastructure Standalone Agent (XAG).

Using XAG automates the mounting of the shared file system (DBFS) as well as the stopping and starting of the Oracle GoldenGate deployment when relocating between Oracle RAC nodes.

Perform the following steps to complete this task:

- Step 8.1 - Modify the Primary Cluster XAG GoldenGate Instance
- Step 8.2 - Install Oracle Grid Infrastructure Agent
- Step 8.3 - Configure Oracle Grid Infrastructure Agent

Step 8.1 - Modify the Primary Cluster XAG GoldenGate Instance
The Oracle Grid Infrastructure Standalone Agent (XAG) GoldenGate instance on the primary system, must be modified to identify that it is part of an Oracle Data Guard configuration.

As the root user on the primary system, use the following command to modify the Oracle Data Guard `autostart` flag:

```
[opc@exapri-node1 ~]$ sudo su -
[root@exapri-node1 ~]# /u01/app/grid/xag/bin/agctl modify goldengate
<instance_name>
 --dataguard_autostart yes
```

Step 8.2 - Install Oracle Grid Infrastructure Agent

On the standby system, follow the instructions in “Step 7.1 - Install the Oracle Grid Infrastructure Standalone Agent” from Task 7 - Configure Oracle Grid Infrastructure Agent.

Step 8.3 - Configure Oracle Grid Infrastructure Agent

The parameters used to register Oracle GoldenGate Microservices with XAG are similar to those used when registering with the primary system.

1. As the `grid` user on the primary system, use the following command to determine the current parameters in the primary system:

```
[grid@exapri-node1 ~]$ agctl config goldengate <instance_name>
Instance name: <instance_name>
Application GoldenGate location is: /u02/app/oracle/goldengate/gg21c
Goldengate MicroServices Architecture environment: yes
Goldengate Service Manager configuration directory:
 /mnt/dbfs/goldengate/deployments/<instance_name>/etc/conf
Goldengate Service Manager var directory:
 /mnt/dbfs/goldengate/deployments/<instance_name>/var
```

```
Service Manager Port: 9100
Goldengate Administration User: oggadmin
Autostart on DataGuard role transition to PRIMARY: yes
Configured to run on Nodes: exapri-node1 exapri-node2
ORACLE_HOME location is: /u02/app/oracle/goldengate/gg21c/lib/instantclient
Database Services needed: ora.<DB_UNIQUE_NAME>.<SERVICE_NAME>.<FQDN>.svc
File System resources needed: dbfs_mount,nginx
Network: 1, IP:<VIP>, User:oracle, Group:oinstall
```

In addition, the XAG parameter `--filesystem_verify no` must be specified to prevent XAG from checking the existence of the DBFS deployment directory when registering the Oracle GoldenGate instance. Without setting this parameter, the XAG registration will fail, because DBFS is not mounted on the standby system.

> **Note:**
>
> It is recommended to use the same GoldenGate instance name when registering GoldenGate with XAG as was used in the primary system.

2. As the `root` user on the standby system, register Oracle GoldenGate Microservices Architecture with XAG use the following command format:

```
https://support.oracle.com/rs?type=doc&id=2193391.1

http://www.oracle.com/pls/topic/lookup?ctx=db19&id=SBYDB

[root@exastb-node1 ~]# /u01/app/grid/xag/bin/agctl add goldengate
<instance_name> \
--gg_home /u02/app/oracle/goldengate/gg21c \
--service_manager \
--config_home /mnt/dbfs/goldengate/deployments/<ggsm1>/etc/conf \
--var_home /mnt/dbfs/goldengate/deployments/<ggsm1>/var \
--port 9100 \
--oracle_home /u02/app/goldengate/gg21c/lib/instantclient \
--adminuser oggadmin \
--user oracle \
--group oinstall \
--network 1 --ip <virtual_IP_address> \
--filesystems dbfs_mount,nginx \
--db_services ora.<DB_UNIQUE_NAME>.<SERVICE_NAME>.<FQDN>.svc \
--use_local_services \
--nodes <exastb-node1>,<exastb-node2> \
--filesystem_verify no \
--dataguard_autostart yes
```

Task 9 - Create Oracle Net TNS Alias for Oracle GoldenGate Database Connections

The same TNS aliases created on the primary system for the primary database using the IPC protocol must be created with the **same** alias names on each database node of the standby

system, using the IPC communication protocol as specified in Cloud: Oracle GoldenGate Microservices Architecture on Oracle Exadata Database Service Configuration Best Practices .

The location of the `tnsnames.ora` used by the Oracle GoldenGate deployment **must** be identical on the standby system nodes as it is on the primary system.

1. As the `oracle` user on the primary system, use the following query REST API call to query the `TNS_ADMIN` location:

```
[opc@exapri-node1 ~]$ sudo su - oracle
[oracle@exapri-node1 ~]$ grep -1 TNS_ADMIN
 /mnt/dbfs/goldengate/deployments/ggsm1/etc/conf/deploymentRegistry.dat
            {
                "name": "TNS_ADMIN",
                "value": "/u02/app/oracle/goldengate/network/admin"
```

   Make sure the `tnsnames.ora` is in this **same** directory on **all** standby database nodes.

2. As the `oracle` OS user on the standby system, follow the steps to create the TNS alias definitions and distribute to all database nodes:

```
[opc@exastb-node1 ~]$ sudo su - oracle
[oracle@exastb-node1 ~]$ dcli -l oracle -g ~/dbs_group mkdir
 -p /u02/app/oracle/goldengate/network/admin
[oracle@exastb-node1 ~]$ vi /u02/app/oracle/goldengate/network/admin/
tnsnames.ora

OGGSRV_CDB =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL=IPC)(KEY=LISTENER))
      (CONNECT_DATA =
        (SERVICE_NAME = <CDB_SERVICE_NAME>)
       )
    )

OGGSRV_<PDB_NAME> =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL=IPC)(KEY=LISTENER))
      (CONNECT_DATA =
        (SERVICE_NAME = <PDB_SERVICE_NAME>)
       )
    )

[oracle@exastb-node1 ~]$ /usr/local/bin/dcli -l oracle -g ~/dbs_group
 -f /u02/app/oracle/goldengate/network/admin/*.ora
 -d /u02/app/oracle/goldengate/network/admin
```

> **Note:**
>
> When the tnsnames.ora or `sqlnet.ora` located in the `TNS_ADMIN` directory for the Oracle GoldenGate deployment are modified, the deployment needs to be restarted in order to pick up the changes.

Task 10 - Configure Oracle GoldenGate Processes

In addition to the advice provided in Cloud: Oracle GoldenGate Microservices Architecture on Oracle Exadata Database Service Configuration Best Practices , follow the recommendations provided below for Extract, Distribution Paths, and Replicats.

Perform the following steps to complete this task:

- Step 10.1 - Modify the Extract Configuration on the Primary System
- Step 10.2 - Modify the Distribution Path Configuration on the Primary and Standby Systems

Step 10.1 - Modify the Extract Configuration on the Primary System

For Extract processes using Data Guard configurations that are using redo transport Maximum Performance or Maximum Availability modes, the following parameter must be added to the Extract process parameter file **on the primary system** to avoid losing transactions and resulting in logical data inconsistencies:

```
TRANLOGOPTIONS HANDLEDLFAILOVER
```

This parameter prevents Extract from extracting transaction data from redo that has not yet been applied to the Data Guard standby database. This is crucial to preventing Oracle GoldenGate from replicating data to a target database that does not exist in the source standby database.

If this parameter is not specified, after a data loss failover of the source database it is possible to have data in the target database that is not present in the source database, leading to logical data inconsistencies.

By default, after 60 seconds, a warning message will be written to the Extract report file when the Extract is stalled due to not being able to query the standby database applied SCN information. For example:

```
WARNING OGG-02721 Extract has been waiting for the standby database for 60
seconds.
```

The amount of time before the warning message is written to Extract report file can be adjusted using the Extract parameter `TRANLOGOPTIONS HANDLEDLFAILOVER STANDBY_WARNING`.

If the Extract is still not able to query the standby database applied SCN information after 30 minutes (default), the Extract process will abend, logging the following message in the Extract report file:

```
ERROR OGG-02722 Extract abended waiting for 1,800 seconds for the standby
database to be accessible or caught up with the primary database.
```

If the standby database becomes available before the default 30 timeout expires, Extract continues mining data from the source database and reports the following message to the report file:

```
INFO OGG-02723 Extract resumed from stalled state and started processing LCRs.
```

The timeout value of 30 minutes can be adjusted using the Extract parameter `TRANLOGOPTIONS HANDLEDLFAILOVER STANDBY_ABEND <value>`, where value is the number of seconds the standby is unavailable before abending.

If the standby database will be unavailable for a prolonged duration, such as during a planned maintenance outage, and you wish Extract to continue extracting data from the primary database, remove the `TRANLOGOPTIONS HANDLEDLFAILOVER` parameter from the Extract parameter file and restart Extract. Remember to set the parameter after the standby becomes available.

> **Note:**
>
> If extracting from a primary database continues while the standby is unavailable, a data loss failover could result after the standby becomes available, and not all the primary redo was applied before a failover. The Oracle GoldenGate target database will contain data that does not exist in the source database.

If the Extract process has been assigned an auto restart profile, as documented in Cloud: Oracle GoldenGate Microservices Architecture on Oracle Exadata Database Service Configuration Best Practices , after a Data Guard role transition, the Extract process will automatically restart. Extract will continue to mine redo data from the new primary database, ignoring the current state of the new standby database, until a default 5-minute timeout period expires. After this time, if the standby is not available Extract will abend with the following errors:

```
INFO OGG-25053 Timeout waiting for 300 seconds for standby database
reinstatement. Now enforcing HANDLEDLFAILOVER.

ERROR OGG-06219 Unable to extract data from the Logmining server OGG$CAP_XXXXX.

ERROR OGG-02078 Extract encountered a fatal error in a processing thread and is
abending.
```

Extract will continue to automatically restart, based on the Oracle GoldenGate Microservices auto restart profile, and failing due to reaching the `HANDLEDLFAILOVER` timeout, until the number retries is reached or the new standby database becomes available.

During the timeout period following a database role transition, the `HANDLEDLFAILOVER` parameter is automatically suspended, so data will be replicated to the Oracle GoldenGate replica database without consideration of the source standby database not being kept up to date. The timeout period for the standby database to start up before Extract abends can be adjusted using the Extract parameter `TRANLOGOPTIONS DLFAILOVER_TIMEOUT`.

It is recommended that you leave `DLFAILOVER_TIMEOUT` at the default of 5 minutes, to allow the old primary to convert to a standby. If the new standby database will be unavailable for an extended period of time or completely gone, then in order for Extract to start and remain running, you must remove the `HANDLEDLFAILOVER` parameter from the Extract parameter file. After removing the parameter, Extract no longer waits until redo has been applied to the standby database before extracting the data.

During the time it takes for the standby database to come back online and apply all the redo from the primary database, there will be data divergence between it and the Oracle GoldenGate replica database. This will be resolved once the standby database is up to date. At which point, add the `HANDLEDLFAILOVER` parameter back into the integrated Extract process parameter file, and then stop and restart the Extract.

When Oracle Data Guard Fast-Start Failover is disabled, such that the broker can automatically fail over to a standby database in the event of loss of the primary database, you must specify an additional integrated Extract parameter shown below.

```
TRANLOGOPTIONS FAILOVERTARGETDESTID n
```

This parameter identifies which standby database the Oracle GoldenGate Extract process must remain behind, with regards to not extracting redo data that has not yet been applied to the standby database.

If Oracle Data Guard Fast-Start Failover is disabled, and you don't specify the additional integrated Extract parameter `FAILOVERTARGETDESTID`, the extract will abend with the following errors:

```
ERROR OGG-06219 Unable to extract data from the Logmining server OGG$CAP_XXXXX.

ERROR OGG-02078 Extract encountered a fatal error in a processing thread and is
abending.
```

To determine the correct value for `FAILOVERTARGETDESTID`, use the `LOG_ARCHIVE_DEST_N` parameter from the Oracle GoldenGate source database which is used for sending redo to the source standby database. For example, if `LOG_ARCHIVE_DEST_2` points to the standby database, then use a value of 2.

As the `oracle` user on the primary system, execute the following command:

```
[opc@exapri-node1 ~]$ sudo su - oracle
[oracle@exapri-node1 ~]$ source <db_name>.env
[oracle@exapri-node1 ~]$ sqlplus / as sysdba

SQL> show parameters log_archive_dest
NAME                 TYPE        VALUE
-------------------- -----------
------------------------------------------------------
log_archive_dest_1   string      location=USE_DB_RECOVERY_FILE_DEST,
                                 valid_for=(ALL_LOGFILES, ALL_ROLES)

log_archive_dest_2   string      service="<db_name>", SYNC AFFIRM delay=0
                                 optional compression=disable max_failure=0
reopen=300
                                 db_unique_name="<db_name>" net_timeout=30,
                                 valid_for=(online_logfile,all_roles)
```

In this example, the Extract parameter would be set to the following:

```
TRANLOGOPTIONS FAILOVERTARGETDESTID 2
```

To add the parameters to the Extract parameter file:

1. Log in into the Oracle GoldenGate **Administration Server** in the Source Oracle GoldenGate.
2. Click in **Overview** under **Administration Service**.
3. Click the **Action** button next to the **Extract** that you want to modify.
4. Select **Details**.
5. Select the **Parameters** tab, and then select the pencil icon to edit the current parameter file.
6. Add the `TRANLOGOPTIONS` parameters and select **Apply** to save the changes.

For the new parameters to take effect, the Extract process needs to be stopped and restarted, which can be done using the Administration Server.

See *Reference for Oracle GoldenGate* for further information about the Extract `TRANLOGOPTIONS` parameters.

Step 10.2 - Modify the Distribution Path Configuration on the Primary and Standby Systems

When the target database of an Oracle GoldenGate environment, where the Receiver Server runs, is protected with Oracle Data Guard, there is an important consideration that must be given to any Distribution Path that are sending trail files to the Receiver Server. When the Receiver Server moves to a different system after an Oracle Data Guard role transition, any Distribution Path must be altered to reflect the new target system address.

You can automatically change the Distribution Paths using a database role transition trigger **in the target database** on the Receiver Server system.

If the primary and standby system VIPs use different root CA certificates, the standby certificate will need to be added to the source deployment Service Manager, as detailed in the "Step 11.3 - Distribution Path Configuration" of Task 11 - Configure Oracle GoldenGate Processes

Follow the instructions below to create a database role transition trigger to modify the Distribution Path target address when the receiver server moves between the primary and standby system, during target database Data Guard role transitions.

Perform the following sub-steps to complete this step:

- Step 10.2.1 - Create a Shell Script to Modify the Distribution Paths
- Step 10.2.2 - Create a DBMS_SCHEDULER job
- Step 10.2.3 - Create the Deployment Config File
- Step 10.2.4 - Create the Database Role Transition Trigger

Step 10.2.1 - Create a Shell Script to Modify the Distribution Paths

Example Distribution Path Target Change Script contains an example shell script that can be used to modify a distribution path target address. Refer to the example script comments for setting appropriate variable values.

> **Note:**
>
> The script should be placed in the same local directory on **all** Oracle RAC nodes of the **TARGETprimary and standby** database systems. Set the script file permissions to 6751.

As the `oracle` OS user on the `TARGET` primary and standby systems, follow the steps to create and distribute the script `change_path_target.sh`:

```
[opc@exadb-node1 ~]$ sudo su - oracle
[oracle@exadb-node1 ~]$ /usr/local/bin/dcli -l oracle -g ~/dbs_group mkdir
 -p /u02/app/oracle/goldengate/scripts
[oracle@exadb-node1 ~]$ vi /u02/app/oracle/goldengate/scripts/
change_path_target.sh
# Paste the script from Example Distribution Path Target Change
    Script
[oracle@exadb-node1 ~]$ /usr/local/bin/dcli -l oracle -g ~/dbs_group
```

```
-f /u02/app/oracle/goldengate/scripts/change_path_target.sh
-d /u02/app/oracle/goldengate/scripts
```

Step 10.2.2 - Create a DBMS_SCHEDULER job

Creating a `DBMS_SCHEDULER` job is required to run an operating system shell script from within PL/SQL.

1. As the `oracle` OS user on the `TARGET` primary system, create the scheduler job as a `SYSDBA` user in the root container database (CDB):

   ```
   [opc@exapri-node1 ~]$ sudo su - oracle
   [oracle@exapri-node1 ~]$ source <db_name>.env
   [oracle@exapri-node1 ~]$ sqlplus / as sysdba
   SQL> exec dbms_scheduler.create_job(job_name=>'gg_change_path_target',
    job_type=>'EXECUTABLE', number_of_arguments => 6,
    job_action=>'/u02/app/oracle/goldengate/scripts/change_path_target.sh',
    enabled=>FALSE);
   ```

   To run an external job, you must set the `run_user` and `run_group` parameters in the `$ORACLE_HOME/rdbms/admin/externaljob.ora` file to the Oracle database operating system user and group.

2. As the `root` OS user on the `TARGET` primary and standby systems, create file `externaljob.ora`:

   ```
   [opc@exadb-node1 ~]$ sudo su -
   [root@exadb-node1 ~]# export DB_NAME=<database_name>
   [root@exadb-node1 ~]# dbaascli database getDetails
    --dbname $DB_NAME |grep homePath |uniq
         "homePath" : "/u02/app/oracle/product/19.0.0.0/dbhome_1",

   [root@exadb-node1 ~]# vi
    /u02/app/oracle/product/19.0.0.0/dbhome_1/rdbms/admin/externaljob.ora
   # Before
   run_user = nobody
   run_group = nobody
   # After
   run_user = oracle
   run_group = oinstall
   ```

3. Repeat this step on all nodes on the primary and standby systems.

> **Note:**
>
> The `extrernaljob.ora` must be configured on all Oracle RAC nodes of the primary and standby database systems.

Step 10.2.3 - Create the Deployment Config File

The example shell script uses REST API calls to access the Oracle GoldenGate distribution path. In order to make the REST API calls secure, it is recommended that you include the user name and password in a configuration file, which is read by `curl`.

As the `oracle` OS user on the `TARGET` primary and standby systems, create the configuration file containing the deployment credentials:

```
[opc@exadb-node1 ~]$ sudo su - oracle
[oracle@exadb-node1 ~]$
cat > /u02/app/oracle/goldengate/scripts/<INSTANCE_NAME>.cfg << EOF
                user = "oggadmin:<password>"
EOF
[oracle@exadb-node1 ~]$ chmod 600 /u02/app/oracle/goldengate/scripts/
<INSTANCE_NAME>.cfg
[oracle@exadb-node1 ~]$ /usr/local/bin/dcli -l oracle -g ~/dbs_group
 -f /u02/app/oracle/goldengate/scripts/<INSTANCE_NAME>.cfg
 -d /u02/app/oracle/goldengate/scripts
```

Step 10.2.4 - Create the Database Role Transition Trigger

Create a role transition trigger on the Oracle GoldenGate source database that will be fire when a standby database becomes a primary database, changing the distribution path target address.

As the `oracle` OS user on the `TARGET` primary system, execute the following SQL sentence to create the role transition trigger:

```
[opc@exapri-node1 ~]$ sudo su - oracle
[oracle@exapri-node1 ~]$ source <db_name>.env
[oracle@exapri-node1 ~]$ sqlplus / as sysdba
CREATE OR REPLACE TRIGGER gg_change_path
AFTER db_role_change ON DATABASE
declare
  role varchar2(30);
  hostname varchar2(64);
begin
  select database_role into role from v$database;
  select host_name into hostname from v$instance;
  DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('gg_change_path_target',1,'<PRIMARY
Source VIP');
  DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('gg_change_path_target',2,'<STANDBY
Source VIP');

DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('gg_change_path_target',4,'<Distribution
 path name');
  DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('gg_change_path_target',5,'<Instance
name>'
  DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('gg_change_path_target',6,'<Config
file containing the deployment credentials>');
  if role = 'PRIMARY' and hostname like '<primary target cluster name>%'
  then
    DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('gg_change_path_target',3,'<PRIMARY
Target VIP>:443');
  elsif role = 'PRIMARY'
  then
    DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('gg_change_path_target',3,'<STANDBY
Target VIP>:443');
  end if;
  DBMS_SCHEDULER.RUN_JOB(job_name=>'gg_change_path_target');
```

```
end;
/
```

Step 10.3 - Replicat Configuration on the Primary System

As documented in “Step 11.4 - Replicat Configuration” of Task 11 - Configure Oracle GoldenGate Processes, a checkpoint table in the target database is required for all Oracle GoldenGate Replicat processes. There are no other configuration requirements for Replicat when configured with Oracle Data Guard.

Example Distribution Path Target Change Script

The following example script can be used to change a source Oracle GoldenGate deployment distribution path target address to reflect the new location of the receiver server after an Oracle Data Guard role transition. This example assumes the source Oracle GoldenGate deployment is configured in an MAA architecture with Data Guard, such that the distribution server can relocate between a primary and standby systems.

```
#!/bin/bash

# change_path_target.sh - changes the target host of a GG Distribution Path
#   when the target moves between primary/standby systems.
# Example usage:
# ./change_path_target.sh <primary source VIP>:443 <standby source VIP>:443
# <path target VIP> <path name> <deployment name> <credentials file>

SOURCE1=$1      # PRIMARY Distribution Server VIP
SOURCE2=$2      # STANDBY Distribution Server VIP
TARGET=$3       # Distribution path target VIP
DPATH=$4        # Distribution path name
DEP=$5          # Deployment name
ACCESS=$6       # Config file containing the deployment credentials.
                # Example contents:
                # user = "oggadmin:<password>"

CONNECT=0

#echo "#${i} - `date`:"
LOGFILE=/tmp/ogg_dpatch_change.txt

result=$(curl -si -K
 $ACCESS https://$SOURCE1/$DEP/distsrvr/services/v2/sources/$DPATH
 -X GET| grep HTTP |  awk '{print $2}')

# Will return NULL of nginx not running, 502 if cannot contact server, 200 if
# contact to server good, and others (404) for other bad reasons:

if [[ -z $result || $result -ne 200 ]]; then  # Managed to access the Distr
Server
  echo "`date` - Couldn't contact Distribution Server at $SOURCE1
 Deployment $DEP ****" >> $LOGFILE
else # Try the other source host:
  echo "`date` - Got status of Distribution Server at $SOURCE1 Deployment
 $DEP ***"  >> $LOGFILE
```

```
  SOURCE=$SOURCE1
  CONNECT=1
fi

if [ $CONNECT -eq 1 ]; then
# For secure NGINX patch destination (wss)
  PAYLOAD='{"target":{"uri":"wss://'${TARGET}'/services/ggnorth/v2/targets?
trail=bb"}}'
  curl -s -K $ACCESS https://$SOURCE/$DEP/distsrvr/services/v2/sources/$DPATH
 -X PATCH --data '{"status": "stopped"}'

# Set new target for path:
  curl -s -K $ACCESS https://$SOURCE/$DEP/distsrvr/services/v2/sources/$DPATH
 -X PATCH --data "$PAYLOAD"
  echo "`date` - Set path $DPATH on $SOURCE deployment $DEP:" >> $LOGFILE

  curl -s -K $ACCESS https://$SOURCE/$DEP/distsrvr/services/v2/sources/$DPATH
 -X GET | python -m json.tool | grep uri >> $LOGFILE
  curl -s -K $ACCESS https://$SOURCE/$DEP/distsrvr/services/v2/sources/$DPATH
 -X PATCH --data '{"status": "running"}'
  exit 0
else
  echo "`date` - ERROR: COULDN'T CHANGE DISTRIBUTION PATH ($DPATH) in
Deployement
 $DEP at $SOURCE! ***" >> $LOGFILE
fi

# If here, means we couldn't connect to either Distribution Servers
exit 1
```

23

On-Premises: Configuring Oracle GoldenGate Hub

Configure and deploy the MAA Oracle GoldenGate Hub architecture using the provided planning considerations, tasks, management, and troubleshooting information.

Topics:

Overview of MAA GoldenGate Hub

To achieve the highest levels of availability, resulting in zero or near-zero downtime for both unplanned outages and planned maintenance activities, customers frequently use the combination of Oracle Real Application Clusters (Oracle RAC), Oracle Active Data Guard, and Oracle GoldenGate.

This architecture, typically referred as Platinum MAA or Never Down Architecture, delivers near zero Recovery Time Objective (RTO, or downtime incurred during outage) and potentially zero or near zero Recovery Point Objective (RPO, or data loss potential).

Traditionally, Oracle GoldenGate is installed and run locally on the database server that the GoldenGate processes connect to. When used with Oracle Grid Infrastructure Standalone Agent (XAG), Oracle GoldenGate processes can be configured to seamlessly relocate or failover between Oracle RAC nodes and follow Oracle Active Data Guard switchover and failovers.

Using MAA Oracle GoldenGate Hub (MAA GGHub) moves the GoldenGate software and processes off of the Exadata database servers, reducing complexity and system resource utilization. MAA GGHub centralizes Oracle GoldenGate management and offloads the majority of the Oracle GoldenGate processing and associated CPU and storage resource utilization from Exadata system resources. Connectivity between the GoldenGate processes and the databases they operate against is managed with Oracle Net Services.

To achieve an MAA Platinum solution on-premises, you follow these high level steps:

1. Review Oracle MAA Platinum Tier for Oracle Exadata to understand Platinum MAA benefits and use cases.
2. Deploy or migrate your database onto Exadata Database Machine
3. Add symmetric standby databases.
4. Configure and deploy Oracle Data Guard Fast Start Failover using the Oracle MAA best practice recommendations in Configure Fast Start Failover.

5. Set up MAA GGHub, which is detailed in the topics that follow.
6. Configure Bidirectional Replication and Automatic Conflict Detection and Resolution.
7. Decide on application failover options such as Global Data Services, or use your own customized application failover.

Refer to the Reference for Oracle GoldenGate for more information about the Extract `TRANLOGOPTIONS` parameters.

When creating an Extract using the Oracle GoldenGate Administration Service GUI, leave the `Trail SubDirectory` parameter blank so that the trail files are automatically created in the deployment directories stored on the shared file system. The default location for trail files is the /<deployment directory>/var/lib/data directory.

> **Note:**
>
> To capture from a multitenant database, you must use an Extract configured at the root level using a c## account. To apply data into a multitenant database, a separate Replicat is needed for each PDB because a Replicat connects at the PDB level and doesn't have access to objects outside of that PDB.

Planning GGHub Placement in the Platinum MAA Architecture

Extreme availability that delivers zero downtime (RTO=0 or near zero) and zero or near zero data loss (RPO=0 or near zero) typically requires the following Platinum MAA architecture where:

1. You have the source and target database in an Oracle GoldenGate architecture to allow your application to fail over immediately in the case of disaster (database, cluster, or site failure) or switch over in the case of a database or application upgrade. This architecture enables the potential RTO of zero or near zero for disaster scenarios and database and application upgrade maintenance.
2. Each source and target database is deployed in Exadata systems so that any local failures are tolerated or recovered almost instantly.
3. Each source and target database is configured with a standby database with Data Guard Fast-Start Failover so that any failure of the database results in activating a new primary database in seconds to minutes. If SYNC transport is leveraged with Maximum Availability protection mode, zero data loss Data Guard failover is achieved.
4. Configured with GoldenGate replication using MAA GGHub between the source and target databases.
5. Configured so that any standby that becomes a primary database because of Data Guard switchover or failover automatically resynchronizes with its target GoldenGate database. If zero data loss Data Guard switchover or failover occurs, GoldenGate resychronization ensures zero data loss across the distributed database environment.
6. Configured with GoldenGate Automatic Conflict Detection and Resolution, which is required after any Data Guard failover operation occurs.

Where to Place the MAA Primary GGHub and Standby GGHub

1. GGHub Pair (Primary and Standby GGhub) must reside in the same region (round trip latency of less 4ms) as each primary and standby database. For example,

a. If the primary database is in Data Center 1, Region A and standby database is in Data Center 2, Region A, then the GGHub pair will reside in Region A.

b. If the primary database is in Region A and standby database is in Region B, then the GGHub pair will split between Region A and B. The primary or active GGhub must be co-located in the same region as the target primary database.

2. Performance implications:

   a. Primary or active GGhub must reside in the same data center as the target database to ensure round trip latency of 4ms or less. (Replicat performance)

   b. Primary or active GGhub should be < 90 ms from the source database without incurring GoldenGate performance degradation (Extract performance).

3. GoldenGate Distribution Path

   a. GoldenGate distribution path is required if the source and target GGhubs are in different areas with a latency > 90 ms.

   b. With bi-directional replication, or when there are multiple target databases in different data centers, it may be necessary to have additional hubs with distribution paths sending trail files between them.

MAA GGHubs Placed in the Same Data Center

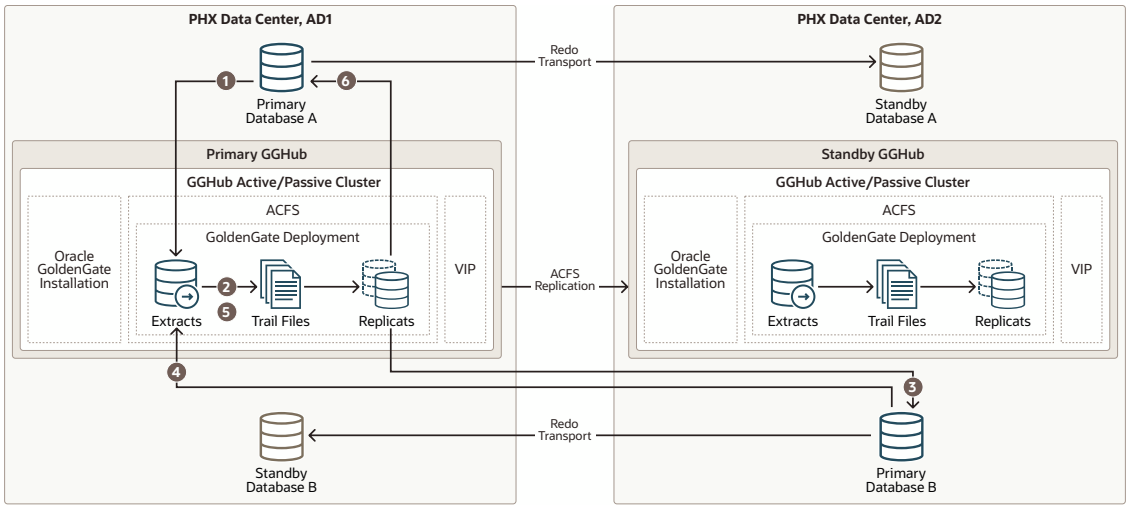
In this scenario, the primary and standby database are located in the same data center (latency less 4ms), and so the primary (active) GGHub and the standby GGHub are also located in the same data center.

The example below has two data centers, or availability domains (ADs), in the same data center.

As shown in Figure 1, you have the following architectural components:

1. Primary database and associated standby database are configured with Oracle Active Data Guard Fast Start Failover (FSFO). FSFO can be configured with Data Guard protection mode with ASYNC or SYNC redo transport depending on your maximum data loss tolerance.
2. Primary GGHub Active/Passive Cluster: Only one GGHub software deployment and configuration on the 2-node cluster. This cluster contains the 21c Oracle GoldenGate software deployment that can support 11.2.0.4 and later database versions. This GGHub can support many primary databases and encapsulates the GoldenGate processes: GoldenGate Extract mines transactions from the source database and GoldenGate Replicat applies the same changes to target database. GoldenGate trail and checkpoint files will also reside in the GGhub ACFS file system. The HA failover solution is built in to the GGhub, which includes automatic failover to the passive node in the same cluster, and restarts GoldenGate processes and activity after a node failure.
3. Standby GGHub Active/Passive Cluster: A Symmetric standby GGhub is configured. ACFS replication is set up between the primary and standby GGHubs to preserve all GoldenGate files. Manual GGhub failover, which includes ACFS failover, can be executed in the rare case that you lose the entire primary GGhub.

Figure 23-1 Primary and Standby GGHubs in the same data center with two separate Availability Domains



The figure above depicts replicating data from Primary Database A to Primary Database B and Primary B back to Primary A with the following steps:

1. Primary Database A: Primary A's Logminer server sends redo changes to a Primary GGHub Extract process.
2. Primary GGHub: An Extract process writes changes to trail files.
3. Primary GGHub to Primary Database B: A Primary GGHub Replicat process applies those changes to the target database (Primary B).
4. Primary Database B: Primary B's Logminer server sends redo to a Primary GGHub Extract process.
5. Primary GGHub: A Primary GGHub Extract process writes changes to trail files.
6. Primary GGHub to Primary Database A: A Primary GGHub Replicat process applies those changes to the target database (Primary A).

Note that one GGHub can support multiple source and target databases, even when the source and target databases are different Oracle Database releases.

Table 23-1 Outage Scenarios, Repair, and Restoring Redundancy for GGHubs in the Same Data Center

Outage Scenario	Application Availability and Repair	Restoring Redundancy and Pristine State
Primary Database A (or Database B) failure	**Impact**: Near-zero application downtime. GoldenGate replication resumes when a new primary database starts. 1. One primary database is still available. All activity is routed to the existing available primary database to achieve zero application downtime. Refer to the Global Data Services Global Services Failover solution. For example, application services A-F are routed to Database A and application services G-J are routed to Database B. If Database A fails, all application services temporarily go to Database B. 2. The standby becomes the new primary automatically with Data Guard FSFO. Oracle GoldenGate replication resumes and the primary databases resynchronize. Data loss is bounded by the Data Guard protection level. If Maximum Availability or Maximum Protection is configured, zero data loss is achieved. All committed transactions are in one or both databases. Workload can be “rebalanced” when Primary Database A and Database B are available and in sync. For example, when Database A is up and running and in sync, services A-F can go back to Database A.	1. The old primary database is reinstated as the new standby database to restore redundancy. 2. Optionally performing a Data Guard switchover to switch back to the original configuration ensures that at least one primary database resides in an independent AD.
Primary or standby GGHub single node failure	**Impact**: No application impact. GoldenGate replication resumes automatically after a couple of minutes. No action is required. The HA failover solution built in to the GGHub includes automatic failover and restart of GoldenGate processes and activity. Replication activity is blocked until GoldenGate processes are active again. GoldenGate replication blackout could last a couple of minutes.	Once the node restarts, active/passive configuration is re-established.

Table 23-1 (Cont.) Outage Scenarios, Repair, and Restoring Redundancy for GGHubs in the Same Data Center

Outage Scenario	Application Availability and Repair	Restoring Redundancy and Pristine State
Primary GGHub cluster crashes and is not recoverable	**Impact**: No application impact. GoldenGate replication resumes after restarting the existing GGHub or executing a manual GGHub failover operation. 1. If the GGHub cluster can be restarted, then that's the simplest solution. 2. If the primary GGHub is not recoverable, then execute a manual GGHub failover to the standby GGHub, which includes ACFS failover. This typically takes several minutes. 3. GoldenGate replication stops until the new primary GGhub is available, so executing step 1 or step 2 should be quick.	If the previous GGHub eventually restarts, ACFS replication resumes in the other direction automatically. If the GGHub cluster is lost or unrecoverable, you need to rebuild a new standby GGHub.
Standby GGHub cluster crashes and not recoverable	**Impact**: No application or replication impact. 1. If the GGHub cluster can be restarted, then that is the simplest solution, and ACFS replication can resume. 2. If the standby GGHub is not recoverable, you can rebuild a new standby GGHub.	N/A

Table 23-1 (Cont.) Outage Scenarios, Repair, and Restoring Redundancy for GGHubs in the Same Data Center

Outage Scenario	Application Availability and Repair	Restoring Redundancy and Pristine State
Complete Data Center or Availability Domain (AD1 or AD2) failure	**Impact**: Near-zero application downtime. GoldenGate replication resumes when the new primary database starts. 1. One primary database is still available. All activity is routed to the existing available primary database to achieve zero application downtime. Refer to the Global Services Failover solution. For example, application services A-F are routed to Database A and application services G-J are routed to Database B. If Database A fails, all services temporarily go to Database B. 2. If the primary GGHub is still functional, GoldenGate replication continues. If the primary GGHub is lost due to availability domain (AD) failure, then a manual GGhub failover is required. GoldenGate replication resumes and the primary databases resynchronize. Data loss is bounded by the Data Guard protection level. If Maximum Availability or Maximum Protection is configured, zero data loss is achieved. All committed transactions are in one or both databases. Workload can be rebalanced when Primary Database A and Database B are available and in sync. When Database A is up and running and in sync, services A-F can go back to Database A.	1. When the data center/AD returns, re-establish configuration such as reinstate standby. If the previous GGHub eventually restarts, ACFS replication resumes in the other direction automatically. 2. When possible, perform a Data Guard switchover (failback) to get back to the original state where one primary database exists in each AD.

MAA GGHubs Placed in Different Data Centers

In this scenario, the primary and standby databases are located in different data centers, and so the primary (active) GGHub is located in the same data center as the primary database, and the standby GGHub is located in the same data center as the standby database.

As shown in the following image, you have the following architectural components:

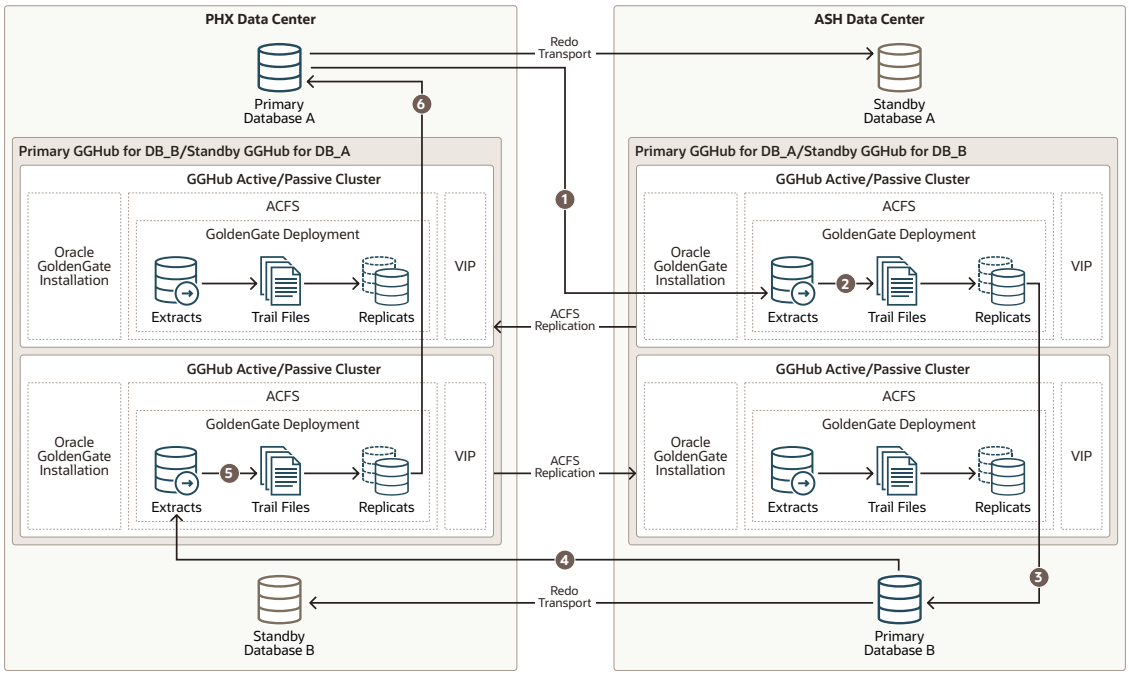
1. The primary database and associated standby database are configured with Oracle Active Data Guard Fast Start Failover (FSFO). FSFO can be configured with Data Guard protection mode with ASYNC or SYNC redo transport depending on your maximum data loss tolerance.

2. Primary GGHub Active/Passive Cluster: In this configuration, there's a 2-node cluster with two Oracle GoldenGate software configurations. Because the primary GGHub needs to be <= 4 ms from the target database and the two data centers network latency > 5 ms, two GGhub configurations are created for each GGHub cluster. Essentially, a primary GGHub configuration is always in the same data center as the target database. GGHub is configured with the Oracle GoldenGate 21c software deployment that can support 11g and later Oracle Database releases. This GGHub can support many primary databases and encapsulates the GoldenGate processes: Extract mines transactions from the source database, and Replicat applies those changes to the target database. GoldenGate trail and checkpoint files will also reside in the ACFS file system. An HA failover solution is built in to the GGhub cluster, which includes automatic failover and restart of GoldenGate processes and activity after a node failure.

   Each GGhub configuration contains a GoldenGate service manager and deployment, ACFS file system with ACFS replication, and a separate application VIP.

3. Standby GGHub Active/Passive Cluster: A symmetric standby GGhub is configured. ACFS replication is set up between the primary and standby GGHubs to preserve all GoldenGate files. Manual GGhub failover, which includes ACFS failover, can be executed if you lose the entire primary GGhub.

Figure 23-2 Primary and Standby GGHubs in Different Data Centers



The figure above depicts replicating data from Primary Database A to Primary Database B and Primary B back to Primary A with the following steps:

1. Primary Database A: Primary A's Logminer server sends redo changes to an PHX DataCenter GGHub Extract process, which is on the Primary GGHub for Database A.
2. Primary GGHub: The Extract process writes changes to trail files.
3. Primary GGHub to Primary Database B: An PHX DataCenter GoldenGate Replicat process applies those changes to the target database (Primary B).
4. Primary Database B: Primary B's Logminer server sends redo to a ASH DataCenter GGHub Extract process, which is on the Primary GGHub for Database B.

5. Primary GGHub: The Extract process writes changes to trail files.
6. Primary GGHub to Primary Database A: A ASH DataCenter GoldenGate Replicat process applies those changes to the target database (Primary A).

Table 23-2 Outage Scenarios, Repair, and Restoring Redundancy for GGHubs in Different Data Centers

Outage Scenario	Application Availability and Repair	Restoring Redundancy and Pristine State
Primary Database A (or Database B) failure	**Impact**: Near-zero application downtime. GoldenGate replication resumes when the new primary database starts. 1. One primary database is still available. All activity is routed to the existing available primary database to achieve zero application downtime. Refer to the Global Data Services Global Services Failover solution. For example, application services A-F are routed to Database A and application services G-J are routed to Database B. If Database A fails, all services temporarily go to Database B. 2. The standby becomes the new primary automatically with Data Guard FSFO. GoldenGate replication resumes and the primary databases resynchronize. Data loss is bounded by the Data Guard protection level. If Maximum Availability or Maximum Protection is configured, zero data loss is achieved. All committed transactions are in one or both databases. Workload can be rebalanced when primary Database A and Database B are available and in sync. For example, when Database A is up and running and in sync, services A-F can go back to Database A. 3. Replicat performance will be degraded if the primary GGHub is not in the same data center as the target database. Schedule a GGHub switchover with ACFS replication switchover to resume optimal Replicat performance to the target database. You may then experience two active GGhub configurations on the same GGHub cluster.	1. The old primary database is reinstated as the new standby database to restore redundancy. 2. Optionally performing a Data Guard switchover, to switch back to the original configuration, ensures that at least one primary database resides in an independent AD. Schedule a GGHub switchover with ACFS replication switchover to resume optimal Replicat performance to the target database.

Table 23-2 (Cont.) Outage Scenarios, Repair, and Restoring Redundancy for GGHubs in Different Data Centers

Outage Scenario	Application Availability and Repair	Restoring Redundancy and Pristine State
Primary or standby GGHub single node failure	**Impact**: No application impact. GoldenGate replication resumes automatically after a couple of minutes. No action is required. An HA failover solution is built in to the GGHub that includes automatic failover and restart of GoldenGate processes and activity. Replication activity is blocked until GoldenGate processes are active again. GoldenGate Replication blackout could last a couple of minutes.	Once the node restarts, active/passive configuration is re-established.
Primary GGHub cluster crashes and is not recoverable	**Impact**: No application impact. GoldenGate replication resumes after the existing primary GGHub restarts or manual GGHub failover completes. 1. If the GGHub cluster can be restarted, then that's the simplest solution. 2. If the primary GGHub is not recoverable, then execute a manual GGHub failover to the standby GGHub, which includes ACFS failover. This typically takes several minutes. 3. Replication stops until the new primary GGhub is started, so executing step 1 or step 2 should be quick. If there's any orchestration, this should be automated.	1. If the previous GGHub eventually restarts, ACFS replication resumes in the other direction automatically. If the GGHub cluster is lost or unrecoverable, you need to rebuild a new standby GGHub. 2. Replicat performance is degraded if the primary GGhub is not in the same data center as the target database. Schedule a GGHub switchover with ACFS replication switchover to resume optimal Replicat performance to the target database.
Standby GGHub cluster crashes and is not recoverable	**Impact**: No application or replication impact. 1. If the GGHub cluster can be restarted, then that's the simplest solution, and ACFS replication will resume. 2. If the standby GGHub is not recoverable, you can rebuild a new standby GGHub.	N/A

Table 23-2 (Cont.) Outage Scenarios, Repair, and Restoring Redundancy for GGHubs in Different Data Centers

Outage Scenario	Application Availability and Repair	Restoring Redundancy and Pristine State
Complete Regional failure	**Impact**: Near Zero Application Downtime. GoldenGate replication resumes once new primary database starts. 1. One primary database is still available. All activity is routed to the existing available primary database to achieve zero application downtime. Refer to the Global Data Services Global Services Failover solution. For example, application services A-F routed to Database A and application services G-J routed to Database B. If Database A fails, all services will temporarily go to Database B. 2. If the primary GGHub is still functional, GoldenGate replication will continue. If the primary GGHub is lost due to regional failure, then a manual GGhub failover is required. GoldenGate replication resumes and the primary databases resynchronize. Data loss is bounded by the Data Guard protection level. If Maximum availability or protection is configured, zero data loss is achieved. All committed transactions are in one or both databases. Workload can be rebalanced when Primary Database A and Database B are available and in sync. When Database A is up and running and in sync, services A-F can go back to Database A.	1. When the data center returns, re-establish configuration such as reinstate standby. If the previous GGHub eventually restarts, ACFS replication resumes in the other direction automatically. 2. When possible, execute a Data Guard switchover (failback) to get back to the original state where one primary database exists in each data center. 3. Replicat performance is degraded if the primary GGHub is not in the same data center as the target database. Schedule a GGHub switchover with ACFS replication switchover to resume optimal Replicat performance to the target database.

Task 1: Configure the Source and Target Databases for Oracle GoldenGate

The source and target Oracle GoldenGate databases should be configured using the following recommendations.

Perform the following steps to complete this task:

- Step 1.1 - Database Configuration
- Step 1.2 - Create the Database Replication Administrator User
- Step 1.3 - Create the Database Services

Step 1.1 - Database Configuration

The source and target Oracle GoldenGate databases should be configured using the following recommendations:

Configuration	Scope	Example

Enable Archivelog Mode	Source and Target	`SQL> ARCHIVE LOG LIST`
		`Database log mode Archive Mode`
		`Automatic archival Enabled`
		`Archive destination USE_DB_RECOVERY_FILE_DEST`
		`Oldest online log sequence 110`
		`Next log sequence to archive 113`
		`Current log sequence 113`

Enable `FORCE LOGGING`	Source and Target	`ALTER DATABASE FORCE LOGGING;`
`ENABLE_GOLDENGATE_REPLICATION`	Source, Target, and Standbys	`ALTER SYSTEM SET ENABLE_GOLDENGATE_REPLICATION=TRUE SCOPE=BOTH SID='*';`
Add Supplemental Logging	Source Required on Target for cases when replication reverses	`ALTER DATABASE ADD SUPPLEMENTAL LOG DATA;`
Configure `STREAMS_POOL_SIZE` larger to accommodate GoldenGate	Source Required on Target for cases when replication reverses	The value of `STREAMS_POOL_SIZE` should be set to the following value: `STREAMS_POOL_SIZE` = (((#Extracts + #Integrated Replicats) * 1GB) * 1.25) For example, in a database with 2 Extracts and 2 integrated Replicats: `STREAMS_POOL_SIZE` = 4GB * 1.25 = 5GB `ALTER SYSTEM SET STREAMS_POOL_SIZE=5G SCOPE=BOTH SID='*';`

For the steps on preparing the database for Oracle GoldenGate, see Preparing the Database for Oracle GoldenGate.

Step 1.2 - Create the Database Replication Administrator User

The source and target databases need a GoldenGate administrator user created, with appropriate privileges assigned as follows:

- For the multitenant container database (CDB):
  - Source database, GoldenGate Extract must be configured to connect to a user in the root container database, using a c##
  - Target database, a separate GoldenGate administrator user is needed for each pluggable database (PDB).
  - For more details about creating a GoldenGate administrator in an Oracle Multitenant Database, see Configuring Oracle GoldenGate in a Multitenant Container Database.
- For non-CDB databases, see Establishing Oracle GoldenGate Credentials

As the `oracle` OS user on the source database system, run the following SQL instructions to create the database user for Oracle GoldenGate and assign the required privileges:

```
[oracle@exadb1_node1 ~]$ sqlplus / as sysdba

# Source CDB
SQL>
alter session set container=cdb$root;
create user c##ggadmin identified by "<ggadmin_password>" container=all
default tablespace USERS temporary tablespace temp;
alter user c##ggadmin quota unlimited on users;
grant set container to c##ggadmin container=all;
grant alter system to c##ggadmin container=all;
grant create session to c##ggadmin container=all;
grant alter any table to c##ggadmin container=all;
grant resource to c##ggadmin container=all;
exec
dbms_goldengate_auth.grant_admin_privilege('c##ggadmin',container=>'all');

# Source PDB
SQL>
alter session set container=<pdbName>;
create user ggadmin identified by "<ggadmin_password>" container=current;
grant create session to ggadmin container=current;
grant alter any table to ggadmin container=current;
grant resource to ggadmin container=current;
exec dbms_goldengate_auth.grant_admin_privilege('ggadmin');
```

As the `oracle` OS user on the target database system, run the following SQL instructions to create the database user for Oracle GoldenGate and assign it the required privileges:

```
[oracle@exadb2_node1 ~]$ sqlplus / as sysdba

# Target PDB
SQL>
alter session set container=<pdbName>;
create user ggadmin identified by "<ggadmin_password>" container=current;
grant alter system to ggadmin container=current;
grant create session to ggadmin container=current;
grant alter any table to ggadmin container=current;
grant resource to ggadmin container=current;
grant dv_goldengate_admin, dv_goldengate_redo_access to ggadmin
container=current;
exec dbms_goldengate_auth.grant_admin_privilege('ggadmin');
```

Step 1.3 - Create the Database Services

If the source and target databases are running the recommended configuration on an Oracle RAC cluster with Oracle Data Guard, a role-based service must be created that allows the Extract or Replicat processes to connect to the correct Data Guard primary database instance.

When using a source multitenant database, a separate service is required for the root container database (CDB) and the pluggable database (PDB) that contains the schema being replicated. For a target multitenant database, a single service is required for the PDB.

As the `oracle` OS user on the primary and standby database systems, create and start the CDB database service using the following command:

```
[oracle@exadb1_node1 ~]$ srvctl add service -db <dbName>
 -service <dbName>_goldengate -preferred <ORACLE_SID1> -available
<ORACLE_SID2>
 -role PRIMARY
```

As the oracle OS user on the primary and standby database systems, create and start the PDB database service using the following command:

```
[oracle@exadb1_node1 ~]$ srvctl add service -db <dbName>
 -service <dbName>_<pdbName>_goldengate -preferred <ORACLE_SID1>
 -available <ORACLE_SID2> -pdb <pdbName> -role PRIMARY
```

As the `oracle` OS user on the primary and standby database systems, start and verify that the services are running, as shown here:

```
[oracle@exadb1_node1 ~]$ srvctl start service -db <dbName> -role
[oracle@exadb1_node1 ~]$ srvctl status service -db <dbName> |grep goldengate

Service <dbName>_goldengate  is running on instance(s) <SID1>
Service <dbName>_<pdbName>_goldengate is running on instance(s) <SID1>
```

> **Note:**
>
> Repeat step 1.3 in the source and target database system.

Task 2: Prepare a Primary and Standby Base System for GGHub

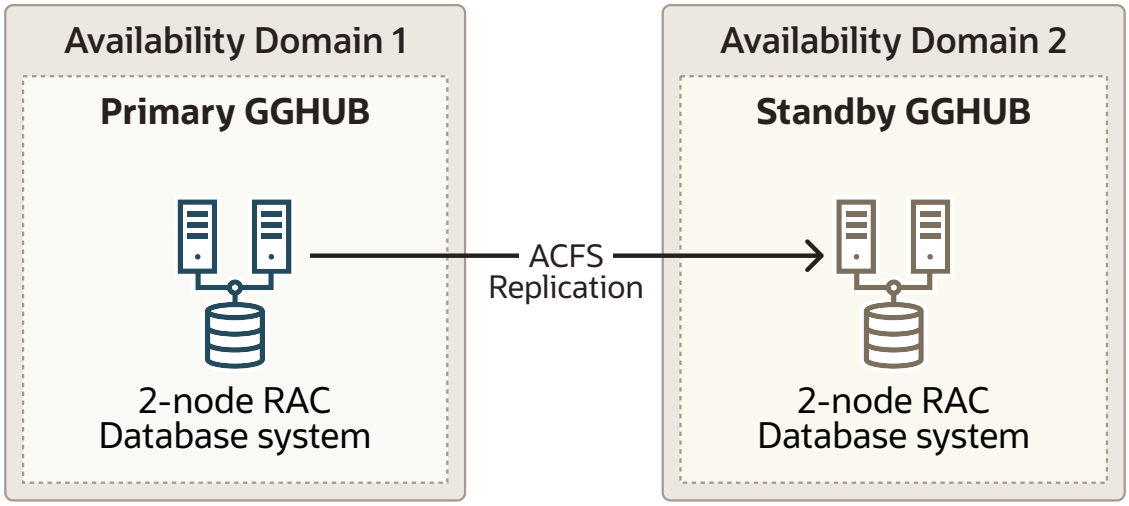
Perform the following steps to complete this task:

- Step 2.1 - Deploy Oracle 2-node Cluster System
- Step 2.2 - Download the Required Software
- Step 2.3 - Configure Oracle Linux to use the Oracle Public YUM Repository

Step 2.1 - Deploy a Oracle 2-Node Oracle Grid Infrastructure System

Deploy a minimum of two GGHubs (primary and standby). Each GGHub must be deployed as a 2-node Oracle Grid Infrastructure system as described in Installing Oracle Grid Infrastructure.

Figure 23-3 Oracle GoldenGate Hub Hardware Architecture



Step 2.2 - Download the Required Software

1. As the `root` OS user on all GGHub nodes, create the staging and scripts directories:

```
[root@gghub_prim1 ~]#
mkdir -p /u01/oracle/stage
mkdir /u01/oracle/scripts
chown -R oracle:oinstall /u01/oracle
chmod -R g+w /u01/oracle
chmod -R o+w /u01/oracle/stage
```

2. As the `opc` OS user on all GGHub nodes, download the following software in the directory /u01/oracle/stage:
   - Download the latest Oracle GoldenGate 21c (or later release) Microservices software from My Oracle Support Doc ID 2193391.1.
   - Download the Oracle Grid Infrastructure Standalone Agents for Oracle Clusterware 19c, release 10.2 or later, from Oracle Grid Infrastructure Standalone Agents for Oracle Clusterware.
   - Download the python script (`secureServices.py`) from My Oracle Support Document 2826001.1
   - Download the Oracle GGHUB Scripts from My Oracle Support Document 2951572.1
3. As the `grid` OS user on all GGHub nodes, unzip the GGhub scripts file downloaded from My Oracle Support Document 2951572.1 into the directory `/u01/oracle/scripts`.

   Place the script in the same location on all primary and standby GGhub nodes

```
[grid@gghub_prim1 ~]$ unzip
 -q /u01/oracle/stage/gghub_scripts_<YYYYYMMDD>.zip
 -d /u01/oracle/scripts/
```

Step 2.3 - Configure Oracle Linux to use the Oracle Public YUM Repository

The Oracle Linux yum server hosts software for Oracle Linux and compatible distributions. These instructions help you get started configuring your Linux system for Oracle Linux yum server and installing software through yum.

For example, as the `root` OS user in all GGHub systems, create the file `/etc/yum.repos.d/oracle-public-yum-ol7.repo` with the following contents:

```
[opc@gghub_prim1 ~]$ sudo su -
[root@gghub_prim1 ~]#
cat > /etc/yum.repos.d/oracle-public-yum-ol7.repo <<EOF
[ol7_latest]
name=Oracle Linux $releasever Latest ($basearch)
baseurl=http://yum.oracle.com/repo/OracleLinux/OL7/latest/\$basearch
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
gpgcheck=1
enabled=1
EOF
```

Task 3: Configure Oracle GoldenGate for the Primary and Standby GGHub

Perform the following steps to complete this task:

- Step 3.1 - Install Oracle GoldenGate Software
- Step 3.2 - Set Up Oracle GoldenGate Hub Architecture Network Configuration
- Step 3.3 - Configure ACFS File System Replication between GGHubs in the same data canter

Step 3.1 - Install Oracle GoldenGate Software

Install Oracle GoldenGate software **locally** on all nodes of the primary and standby GGHub configuration that will be part of the GoldenGate configuration. Make sure the installation directory is identical on all nodes.

Step 3.1.1 Unzip the Software and Create the Response File for the Installation

As the `oracle` OS user on all GGHub nodes, unzip the Oracle GoldenGate software:

```
[oracle@gghub_prim1 ~]$ unzip -q
 /u01/oracle/stage/p36175132_21130000GGRU_Linux-x86-64.zip -d /u01/oracle/
stage
```

The software includes an example response file for Oracle Database 21c and earlier supported versions.

Copy the response file to a shared file system, so the same file can be used to install Oracle GoldenGate on all database nodes, and edit the following parameters:

- `INSTALL_OPTION=ora21c`
- `SOFTWARE_LOCATION=/u01/app/oracle/goldengate/gg21c (recommended location)`

As the `oracle` OS user on all GGHub nodes, copy and edit the response file for the installation:

```
[oracle@gghub_prim1 ~]$ cp
 /u01/oracle/stage/fbo_ggs_Linux_x64_Oracle_services_shiphome/Disk1/response/
oggcore.rsp
 /u01/oracle/stage
[oracle@gghub_prim1 ~]$ vi /u01/oracle/stage/oggcore.rsp

# Before
INSTALL_OPTION=
SOFTWARE_LOCATION=

# After
INSTALL_OPTION=ora21c
SOFTWARE_LOCATION=/u01/app/oracle/goldengate/gg21c
```

Step 3.1.2 Install Oracle GoldenGate Software

As the `oracle` OS user on all GGHub nodes, run `runInstaller` to install Oracle GoldenGate:

```
[oracle@gghub_prim1 ~]$ cd
 /u01/oracle/stage/fbo_ggs_Linux_x64_Oracle_services_shiphome/Disk1/
[oracle@gghub_prim1 ~]$ ./runInstaller -silent -nowait
 -responseFile /u01/oracle/stage/oggcore.rsp

Starting Oracle Universal Installer...

Checking Temp space: must be greater than 120 MB.   Actual 32755 MB    Passed
Checking swap space: must be greater than 150 MB.   Actual 16383 MB    Passed
Preparing to launch Oracle Universal Installer from
 /tmp/OraInstall2022-07-08_02-54-51PM. Please wait ...
You can find the log of this install session at:
 /u01/app/oraInventory/logs/installActions2022-07-08_02-54-51PM.log
Successfully Setup Software.
The installation of Oracle GoldenGate Services was successful.
Please check '/u01/app/oraInventory/logs/
silentInstall2022-07-08_02-54-51PM.log'
 for more details.

[oracle@gghub_prim1 ~]$ cat
 /u01/app/oraInventory/logs/silentInstall2022-07-08_02-54-51PM.log

The installation of Oracle GoldenGate Services was successful.
```

Step 3.1.3 Install Patches for Oracle GoldenGate Microservices Architecture

As the `oracle` OS user on all GGHub nodes, install the latest OPatch:

```
[oracle@gghub_prim1 ~]$ unzip -oq
 -d /u01/app/oracle/goldengate/gg21c
 /u01/oracle/stage/p6880880_210000_Linux-x86-64.zip
[oracle@gghub_prim1 ~]$ cat >> ~/.bashrc <<EOF
export ORACLE_HOME=/u01/app/oracle/goldengate/gg21c
export PATH=/u01/app/oracle/goldengate/gg21c/OPatch:$PATH
EOF
```

```
[oracle@gghub_prim1 ~]$ . ~/.bashrc
[oracle@gghub_prim1 ~]$ opatch lsinventory |grep 'Oracle GoldenGate Services'

Oracle GoldenGate Services
21.1.0.0.0

[oracle@gghub_prim1 Disk1]$ opatch version
OPatch Version: 12.2.0.1.37

OPatch succeeded.
```

As the `oracle` OS user on all GGHub nodes, run OPatch `prereq` to validate any conflict before applying the patch:

```
[oracle@gghub_prim1 ~]$ unzip -oq
 -d /u01/oracle/stage/ /u01/oracle/stage/p35214851_219000OGGRU_Linux-
x86-64.zip

[oracle@gghub_prim1 ~]$ cd /u01/oracle/stage/35214851/
[oracle@gghub_prim1 35214851]$ opatch prereq CheckConflictAgainstOHWithDetail
-ph ./

Oracle Interim Patch Installer version 12.2.0.1.26
Copyright (c) 2023, Oracle Corporation.  All rights reserved.

PREREQ session

Oracle Home       : /u01/app/oracle/goldengate/gg21c
Central Inventory : /u01/app/oraInventory
   from           : /u01/app/oracle/goldengate/gg21c/oraInst.loc
OPatch version    : 12.2.0.1.26
OUI version       : 12.2.0.9.0
Log file location :
 /u01/app/oracle/goldengate/gg21c/cfgtoollogs/opatch/
opatch2023-04-21_13-44-16PM_1.log

Invoking prereq "checkconflictagainstohwithdetail"

Prereq "checkConflictAgainstOHWithDetail" passed.

OPatch succeeded.
```

As the `oracle` OS user on all GGHub nodes, patch Oracle GoldenGate Microservices Architecture using OPatch:

```
[oracle@gghub_prim1 35214851]$ opatch apply

Oracle Interim Patch Installer version 12.2.0.1.37
Copyright (c) 2023, Oracle Corporation.  All rights reserved.

Oracle Home       : /u01/app/oracle/goldengate/gg21c
Central Inventory : /u01/app/oraInventory
   from           : /u01/app/oracle/goldengate/gg21c/oraInst.loc
OPatch version    : 12.2.0.1.37
OUI version       : 12.2.0.9.0
```

```
Log file location :
 /u01/app/oracle/goldengate/gg21c/cfgtoollogs/opatch/
opatch2023-04-21_19-40-41PM_1.log
Verifying environment and performing prerequisite checks...
OPatch continues with these patches:   35214851

Do you want to proceed? [y|n]
y
User Responded with: Y
All checks passed.

Please shutdown Oracle instances running out of this ORACLE_HOME on the local
system.
(Oracle Home = '/u01/app/oracle/goldengate/gg21c')

Is the local system ready for patching? [y|n]
y
User Responded with: Y
Backing up files...
Applying interim patch '35214851' to OH '/u01/app/oracle/goldengate/gg21c'

Patching component oracle.oggcore.services.ora21c, 21.1.0.0.0...
Patch 35214851 successfully applied.
Log file location:
 /u01/app/oracle/goldengate/gg21c/cfgtoollogs/opatch/
opatch2023-04-21_19-40-41PM_1.log

OPatch succeeded.

[oracle@gghub_prim1 35214851]$ opatch lspatches
35214851;

OPatch succeeded.
```

> **Note:**
>
> Repeat all of the sub steps in step 3.1 for the primary and standby GGHub systems.

Step 3.2 - Create Application Virtual IP Address (VIP)

A dedicated application virtual IP address (VIP) is required on each hub cluster to ensure that the primary ACFS replication process sends file system data to the correct hub standby node where the file system is currently mounted. This is accomplished by co-locating the VIP and the ACFS CRS resources on the same node. The VIP is a cluster resource that Oracle Clusterware manages, and is migrated to another cluster node in the event of a node failure.

As the `root` OS user on the first GGHub node, run the following command to identify the network number:

```
[root@gghub_prim1 ~]# $(grep ^crs_home /etc/oracle/olr.loc | cut -d=
 -f2)/bin/crsctl status resource -p -attr NAME,USR_ORA_SUBNET
 -w "TYPE = ora.network.type" |sort | uniq
```

```
NAME=ora.net1.network
USR_ORA_SUBNET=10.128.26.0
```

As the `root` OS user on the first GGHub node, run the following command to create the application VIP managed by Oracle Clusterware:

```
[root@gghub_prim1 ~]# sh /u01/oracle/scripts/add_appvip.sh

Application VIP Name: gghub_prim_vip1
Application VIP Address: 10.128.26.200
Using configuration parameter file: /u01/app/19.0.0.0/grid/crs/install/
crsconfig_params
The log of current session can be found at:
  /u01/app/grid/crsdata/gghub_prim1/scripts/appvipcfg.log
```

Step 3.3 - Configure ACFS File System Replication between GGHubs in the Same Region

Oracle GoldenGate Microservices Architecture is designed with a simplified installation and deployment directory structure. The installation directory should be placed on local storage on each database node to minimize downtime during software patching. The deployment directory, which is created during deployment creation using the Oracle GoldenGate Configuration Assistant (oggca.sh), must be placed on a shared file system.

The deployment directory contains configuration, security, log, parameter, trail, and checkpoint files. Placing the deployment in Oracle Automatic Storage Management Cluster File system (ACFS) provides the best recoverability and failover capabilities in the event of a system failure. Ensuring the availability of the checkpoint files cluster-wide is essential so that the GoldenGate processes can continue running from their last known position after a failure occurs.

It is recommended that you allocate enough trail file disk space for a minimum of 12 hours of trail files. This will provide sufficient space for trail file generation should a problem occur with the target environment that prevents it from receiving new trail files. The amount of space needed for 12 hours can only be determined by testing trail file generation rates with real production data.

If you want to build contingency for a long planned maintenance event of one of the GoldenGate Primary Database or systems, you can allocate sufficient ACFS space for 2 days. Monitoring space utilization is always recommended regardless of how much space is allocated.

> **Note:**
>
> If the GoldenGate hub will support multiple service manager deployments using separate ACFS file systems, the following steps should be repeated for each file ACFS file system.

Perform the following sub-steps to complete this step:

- Step 3.3.1 - Create the ASM File system
- Step 3.3.2 - Create the Cluster Ready Services (CRS) Resource
- Step 3.3.3 - Verify the Currently Configured ACFS File System

- Step 3.3.4 - Start and Check the Status of the ACFS Resource
- Step 3.3.5 – Create CRS Dependencies Between ACFS and an Application VIP
- Step 3.3.6 – Create the SSH Daemon CRS Resource
- Step 3.3.7 – Enable ACFS Replication
- Step 3.3.8 – Create the ACFS Replication CRS Action Scripts

Step 3.3.1 - Create the ASM File system

As the `grid` OS user on the first GGHub node, use `asmcmd` to create the ACFS volume:

```
[grid@gghub_prim1 ~]$ asmcmd volcreate -G DATA -s 120G ACFS_GG1
```

> **Note:**
>
> Modify the file system size according to the determined size requirements.

As the `grid` OS user on the first GGHub node, use `asmcmd` to confirm the “Volume Device”:

```
[grid@gghub_prim1 ~]$ asmcmd volinfo -G DATA ACFS_GG1

Diskgroup Name: DATA
         Volume Name: ACFS_GG1
         Volume Device: /dev/asm/acfs_gg1-256
         State: ENABLED
         Size (MB): 1228800
         Resize Unit (MB): 64
         Redundancy: UNPROT
         Stripe Columns: 8
         Stripe Width (K): 1024
         Usage:
         Mountpath:
```

As the `grid` OS user on the first GGHub node, format the partition with the following `mkfs` command:

```
[grid@gghub_prim1 ~]$ /sbin/mkfs -t acfs /dev/asm/acfs_gg1-256

mkfs.acfs: version                   = 19.0.0.0.0
mkfs.acfs: on-disk version           = 46.0
mkfs.acfs: volume                    = /dev/asm/acfs_gg1-256
mkfs.acfs: volume size               = 128849018880  ( 120.00 GB )
mkfs.acfs: Format complete.
```

Step 3.3.2 - Create the Cluster Ready Services (CRS) Resource

As the `root` OS user on all GGHub nodes, create the ACFS mount point:

```
[root@gghub_prim1 ~]# mkdir -p /mnt/acfs_gg1
[root@gghub_prim1 ~]# chown oracle:oinstall /mnt/acfs_gg1
```

Create the file system resource as the `root` user. Due to the implementation of distributed file locking on ACFS, unlike DBFS, it is acceptable to mount ACFS on more than one GGHub node at any one time.

As the `root` OS user on the first GGHub node, create the CRS resource for the new ACFS file system:

```
[root@gghub_prim1 ~]# vi /u01/oracle/scripts/add_asm_filesystem.sh
# Run as ROOT
$(grep ^crs_home /etc/oracle/olr.loc | cut -d= -f2)/bin/srvctl add filesystem
\
-device /dev/asm/<acfs_volume> \
-volume ACFS_GG1 \
-diskgroup DATA \
-path /mnt/acfs_gg1 -user oracle \
-node gghub_prim1,gghub_prim2 \
-autostart NEVER \
-mountowner oracle \
-mountgroup oinstall \
-mountperm 755

[root@gghub_prim1 ~]# sh /u01/oracle/scripts/add_asm_filesystem.sh
```

Step 3.3.3 - Verify the Currently Configured ACFS File System

As the `grid` OS user on the first GGHub node, use the following command to validate the file system details:

```
[grid@gghub_prim1 ~]$ srvctl config filesystem -volume ACFS_GG1
 -diskgroup DATA

Volume device: /dev/asm/acfs_gg1-256
Diskgroup name: data
Volume name: acfs_gg1
Canonical volume device: /dev/asm/acfs_gg1-256
Accelerator volume devices:
Mountpoint path: /mnt/acfs_gg1
Mount point owner: oracle
Mount point group: oinstall
Mount permissions: owner:oracle:rwx,pgrp:oinstall:r-x,other::r-x
Mount users: grid
Type: ACFS
Mount options:
Description:
Nodes: gghub_prim1 gghub_prim2
Server pools: *
Application ID:
ACFS file system is enabled
ACFS file system is individually enabled on nodes:
ACFS file system is individually disabled on nodes:
```

Step 3.3.4 - Start and Check the Status of the ACFS Resource

As the `grid` OS user on the first GGHub node, use the following command to start and check the file system:

```
[grid@gghub_prim1 ~]$ srvctl start  filesystem -volume ACFS_GG1
 -diskgroup DATA -node `hostname`
[grid@gghub_prim1 ~]$ srvctl status filesystem -volume ACFS_GG1 -diskgroup
DATA
ACFS file system /mnt/acfs_gg1 is mounted on nodes gghub_prim1
```

The CRS resource created is named using the format `ora.diskgroup_name.volume_name.acfs`. Using the above file system example, the CRS resource is called `ora.data.acfs_gg.acfs`.

As the `grid` OS user on the first GGHub node, use the following command to see the ACFS resource in CRS:

```
[grid@gghub_prim1 ~]$ crsctl stat res ora.data.acfs_gg1.acfs

NAME=ora.data.acfs_gg1.acfs
TYPE=ora.acfs_cluster.type
TARGET=ONLINE
STATE=ONLINE on gghub_prim1
```

Step 3.3.5 – Create CRS Dependencies Between ACFS and an Application VIP

To ensure that the file system is mounted on the same Oracle GGHub node as the VIP, add the VIP CRS resource as a dependency to the ACFS resource, using the following example commands. Each separate replicated ACFS file system will have its own dedicated VIP.

As the `root` OS user on the first GGHub node, use the following command to determine the current start and stop dependencies of the VIP resource:

```
[root@gghub_prim1 ~]# export APPVIP=`$(grep ^crs_home /etc/oracle/olr.loc |
cut
 -d= -f2)/bin/crsctl stat res -w "TYPE co appvip" |grep NAME | cut -f2 -d"="`
gghub_prim_vip1

[root@gghub_prim1 ~]# export APPVIP=gghub_prim_vip1
[root@gghub_prim1 ~]# $(grep ^crs_home /etc/oracle/olr.loc | cut -d= -f2)/bin/
crsctl
 stat res $APPVIP -f|grep _DEPENDENCIES

START_DEPENDENCIES=hard(ora.net1.network) pullup(ora.net1.network)
STOP_DEPENDENCIES=hard(intermediate:ora.net1.network)
```

As the `root` OS user on the first GGHub node, determine the ACFS file system name:

```
[root@gghub_prim1 ~]# $(grep ^crs_home /etc/oracle/olr.loc | cut -d=
 -f2)/bin/crsctl stat res -w "NAME co acfs_gg1" |grep NAME

NAME=ora.data.acfs_gg.acfs

[root@gghub_prim1 ~]# export ACFS_NAME=ora.data.acfs_gg1.acfs
```

As the `root` OS user on the first GGHub node, modify the start and stop dependencies of the VIP resource:

```
[root@gghub_prim1 ~]# $(grep ^crs_home /etc/oracle/olr.loc | cut -d=
 -f2)/bin/crsctl modify res $APPVIP
 -attr "START_DEPENDENCIES='hard(ora.net1.network,$ACFS_NAME)
pullup(ora.net1.network)

pullup:always($ACFS_NAME)',STOP_DEPENDENCIES='hard(intermediate:ora.net1.netwo
rk,$ACFS_NAME)',HOSTING_MEMBERS=,PLACEMENT=balanced"
```

As the `grid` OS user on the first GGHub node, start the VIP resource:

```
[grid@gghub_prim1 ~]$ $(grep ^crs_home /etc/oracle/olr.loc | cut -d=
 -f2)/bin/crsctl stat res -w "TYPE co appvip" |grep NAME | cut -f2 -d"="
gghub_prim_vip1

[grid@gghub_prim1 ~]$ export APPVIP=gghub_prim_vip1


[grid@gghub_prim1 ~]$ crsctl start resource $APPVIP

CRS-2672: Attempting to start 'gghub_prim_vip1' on 'gghub_prim1'
CRS-2676: Start of 'gghub_prim_vip1' on 'gghub_prim1' succeeded
```

> **Note:**
>
> Before moving to the next step, it is important to make sure that the VIP can be mounted on both GGHub nodes.

As the `grid` OS user on the first GGHub node, relocate the VIP resource:

```
[grid@gghub_prim1 ~]$ crsctl relocate resource $APPVIP -f

CRS-2673: Attempting to stop 'gghub_prim_vip1' on 'gghub_prim1'
CRS-2677: Stop of 'gghub_prim_vip1' on 'gghub_prim1' succeeded
CRS-2673: Attempting to stop 'ora.data.acfs_gg1.acfs' on 'gghub_prim1'
CRS-2677: Stop of 'ora.data.acfs_gg1.acfs' on 'gghub_prim1' succeeded
CRS-2672: Attempting to start 'ora.data.acfs_gg1.acfs' on 'gghub_prim2'
CRS-2676: Start of 'ora.data.acfs_gg1.acfs' on 'gghub_prim2' succeeded
CRS-2672: Attempting to start 'gghub_prim_vip1' on 'gghub_prim2'
CRS-2676: Start of 'gghub_prim_vip1' on 'gghub_prim2' succeeded

[grid@gghub_prim1 ~]$ crsctl status resource $APPVIP

NAME=gghub_prim_vip1
TYPE=app.appviptypex2.type
TARGET=ONLINE
STATE=ONLINE on gghub_prim2

[grid@gghub_prim1 ~]$ crsctl relocate resource $APPVIP -f
```

```
CRS-2673: Attempting to stop 'gghub_prim_vip1' on 'gghub_prim2'
CRS-2677: Stop of 'gghub_prim_vip1' on 'gghub_prim2' succeeded
CRS-2673: Attempting to stop 'ora.data.acfs_gg1.acfs' on 'gghub_prim2'
CRS-2677: Stop of 'ora.data.acfs_gg1.acfs' on 'gghub_prim2' succeeded
CRS-2672: Attempting to start 'ora.data.acfs_gg1.acfs' on 'gghub_prim1'
CRS-2676: Start of 'ora.data.acfs_gg1.acfs' on 'gghub_prim1' succeeded
CRS-2672: Attempting to start 'gghub_prim_vip1' on 'gghub_prim1'
CRS-2676: Start of 'gghub_prim_vip1' on 'gghub_prim1' succeeded
```

As the `grid` OS user on the first GGHub node, check the status of the ACFS file system:

```
[grid@gghub_prim1 ~]$ srvctl status filesystem
 -volume ACFS_GG1 -diskgroup DATA

ACFS file system /mnt/acfs_gg1 is mounted on nodes gghub_prim1
```

Step 3.3.6 – Create the SSH Daemon CRS Resource

ACFS replication uses a secure shell (ssh) to communicate between the primary and standby file systems using the virtual IP addresses that were previously created. When a server is rebooted, the ssh daemon is started before the VIP CRS resource, preventing access to the cluster using VIP.

The following instructions create a ssh restart CRS resource that will restart the ssh daemon after the virtual IP resource is started. A separate ssh restart CRS resource is needed for each replicated file system.

As the `root` OS user on the first GGHub node, create the CRS resource using the following command:

```
[root@gghub_prim1 ~]# $(grep ^crs_home /etc/oracle/olr.loc | cut
 -d= -f2)/bin/crsctl stat res -w "TYPE co appvip" |grep NAME | cut -f2 -d"="
gghub_prim_vip1

[root@gghub_prim1 ~]# export APPVIP=gghub_prim_vip1
[root@gghub_prim1 ~]# sh /u01/oracle/scripts/add_sshd_restart.sh
```

As the `grid` OS user on the first GGHub node, start and test the CRS resource:

```
[grid@gghub_prim1 ~]$ crsctl stat res sshd_restart
NAME=sshd_restart
TYPE=cluster_resource
TARGET=OFFLINE
STATE=OFFLINE

[grid@gghub_prim1 ~]$ crsctl start res sshd_restart

CRS-2672: Attempting to start 'sshd_restart' on 'gghub_prim1'
CRS-2676: Start of 'sshd_restart' on 'gghub_prim1' succeeded
[grid@gghub_prim1 ~]$ cat /tmp/sshd_restarted
STARTED

[grid@gghubtest1 ~]$ crsctl stop res sshd_restart

CRS-2673: Attempting to stop 'sshd_restart' on 'gghub_prim1'
```

```
CRS-2677: Stop of 'sshd_restart' on 'gghub_prim1' succeeded
[grid@gghub1 ~]$ cat /tmp/sshd_restarted
STOPPED

[grid@gghub1 ~]$ crsctl start res sshd_restart

CRS-2672: Attempting to start 'sshd_restart' on 'gghub_prim1'
CRS-2676: Start of 'sshd_restart' on 'gghub_prim1' succeeded
[grid@gghub1 ~]$ crsctl stat res sshd_restart

NAME=sshd_restart
TYPE=cluster_resource
TARGET=ONLINE
STATE=ONLINE on gghub_prim1
```

Step 3.3.7 – Enable ACFS Replication

ACFS snapshot-based replication uses `openssh` to transfer the snapshots from between the primary and standby hosts using the designated replication user, which is commonly the `grid` user.

As the `grid` OS user in the primary and standby hub systems, follow the instructions in Configuring ssh for Use With Oracle ACFS Replication to configure the ssh connectivity between the primary and standby nodes.

As the `grid` OS user on all primary and standby GGHub nodes, use `ssh` to test connectivity between all primary to standby nodes, and in the reverse direction using ssh as the replication user:

```
# On the Primary GGhub
[grid@gghub_prim1 ~]$ ssh gghub_stby_vip1.frankfurt.goldengate.com hostname
gghub_stby1

[grid@gghub_prim2 ~]$ ssh gghub_stby_vip1.frankfurt.goldengate.com hostname
gghub_stby1

# On the Standby GGhub

[grid@gghub_stby1 ~]$ ssh gghub_prim_vip1.frankfurt.goldengate.com hostname
gghub_prim1

[grid@gghub_stby2 ~]$ ssh gghub_prim_vip1.frankfurt.goldengate.com hostname
gghub_prim1
```

As the `grid` OS user on the primary and standby GGHub nodes where ACFS is mounted, use `acfsutil` to test connectivity between the primary and the standby nodes:

```
# On the Primary GGhub

[grid@gghub_prim1 ~]$ srvctl status filesystem -volume ACFS_GG1 -diskgroup
DATA
ACFS file system /mnt/acfs_gg1 is mounted on nodes gghub_prim1

[grid@gghub_prim1 ~]$ acfsutil repl info -c
-u grid gghub_prim_vip1.frankfurt.goldengate.com
```

```
 gghub_stby_vip1.frankfurt.goldengate.com
 /mnt/acfs_gg1
A valid 'ssh' connection was detected for standby node
 gghub_prim_vip1.frankfurt.goldengate.com as user grid.
A valid 'ssh' connection was detected for standby node
 gghub_stby_vip1.frankfurt.goldengate.com as user grid.

# On the Standby GGhub


[grid@gghub_stby1 ~]$ srvctl status filesystem -volume ACFS_GG1
 -diskgroup DATA
ACFS file system /mnt/acfs_gg1 is mounted on nodes gghub_stby1


[grid@gghub_stby1 ~]$ acfsutil repl info -c
 -u grid gghub_prim_vip1.frankfurt.goldengate.com
 gghub_stby_vip1.frankfurt.goldengate.com
 /mnt/acfs_gg
A valid 'ssh' connection was detected for standby node
 gghub_prim_vip1.frankfurt.goldengate.com as user grid.
A valid 'ssh' connection was detected for standby node
 gghub_stby_vip1.frankfurt.goldengate.com as user grid.
```

If the `acfsutil` command is run from a GGHub node where ACFS is not mounted, the error ACFS-05518 will be shown as expected.

Use `srvctl status filesytem` to find the GGHub where ACFS is mounted and re-run the command:

```
[grid@gghub_prim1 ~]$ acfsutil repl info -c
 -u grid gghub_stby_vip1.frankfurt.goldengate.com
gghub_stby_vip1.frankfurt.goldengate.com
 /mnt/acfs_gg1
acfsutil repl info: ACFS-05518: /mnt/acfs_gg1 is not an ACFS mount point

[grid@gghub_prim1 ~]$ srvctl status filesystem -volume ACFS_GG1 -diskgroup
DATA
ACFS file system /mnt/acfs_gg1 is mounted on nodes gghub_prim2

[grid@gghub_prim1 ~]$ ssh gghub_prim2
[grid@gghub_prim2 ~]$ acfsutil repl info -c -u
 grid gghub_prim_vip1.frankfurt.goldengate.com
gghub_stby_vip1.frankfurt.goldengate.com
 /mnt/acfs_gg1
A valid 'ssh' connection was detected for standby node
 gghub_prim_vip1.frankfurt.goldengate.com as user grid.
A valid 'ssh' connection was detected for standby node
 gghub_stby_vip1.frankfurt.goldengate.com as user grid.
```

> **Note:**
>
> Make sure the connectivity is verified between all primary nodes to all standby nodes, as well as in the opposite direction. Only continue when there are no errors with any of the connection tests.

As the `grid` OS user on the standby GGhub node where ACFS is currently mounted, initialize ACFS replication:

```
[grid@gghub_stby1 ~]$ srvctl status filesystem -volume ACFS_GG1 -diskgroup
DATA

ACFS file system /mnt/acfs_gg1 is mounted on nodes gghub_stby1

[grid@gghub_stby1 ~]$ /sbin/acfsutil repl init standby -u grid /mnt/acfs_gg1
```

As the `grid` OS user on the primary GGhub node where ACFS is currently mounted, initialize ACFS replication:

```
[grid@gghub_prim1 ~]$ srvctl status filesystem -volume ACFS_GG1 -diskgroup
DATA

ACFS file system /mnt/acfs_gg is mounted on nodes gghub_prim1

[grid@gghub_prim1 ~]$ /sbin/acfsutil repl init primary -C -p
 grid@gghub_prim_vip1.frankfurt.goldengate.com -s
 grid@gghub_stby_vip1.frankfurt.goldengate.com -m /mnt/acfs_gg1 /mnt/acfs_gg1
```

As the `grid` OS user on the primary and standby GGhub nodes, monitor the initialization progress.

When the status changes to “Send Completed” it means that the initial primary file system copy has finished and the primary file system is now being replicated to the standby host:

```
# On the Primary GGhub

[grid@gghub_prim1 ~]$ /sbin/acfsutil repl info -c -v /mnt/acfs_gg1 | grep
Status

Status:                               Send Completed

# On the Standby GGhub

[grid@gghub_prim1 ~]$ /sbin/acfsutil repl info -c -v /mnt/acfs_gg1 | grep
Status

Status:                               Receive Completed
```

As the `grid` OS user on the primary and standby GGhub nodes, verify and monitor the ACFS replicated file system:

```
# On the Primary GGhub

[grid@gghub_prim1 ~]$ acfsutil repl util verifystandby /mnt/acfs_gg1

verifystandby returned: 0

# On the Standby GGhub

[grid@gghubtest31 ~]$ acfsutil repl util verifyprimary /mnt/acfs_gg1

verifyprimary returned: 0
```

> **Note:**
>
> Both commands will return a value of 0 (zero) if there are no problems detected. See Troubleshooting ACFS Replication for monitoring, diagnosing, and resolving common issues with ACFS Replication before continuing.

As the `grid` OS user on the primary GGhub node, use the following command to monitor the status of the ACFS replication:

```
[grid@gghub_prim1 ~]$ /sbin/acfsutil repl info -c -v /mnt/acfs_gg1

Site:                                 Primary
Primary hostname:                     gghub_prim_vip1.frankfurt.goldengate.com
Primary path:                         /mnt/acfs_gg1
Primary status:                       Running
Background Resources:                 Active

Standby connect string:
grid@gghub_stby_vip1.frankfurt.goldengate.com
Standby path:                         /mnt/acfs_gg1
Replication interval:                 0 days, 0 hours, 0 minutes, 0 seconds
Sending primary as of:                Fri May 05 12:37:02 2023
Status:                               Send Completed
Lag Time:                             00:00:00
Retries made:                         0
Last send started at:                 Fri May 05 12:37:02 2023
Last send completed at:               Fri May 05 12:37:12 2023
Elapsed time for last send:           0 days, 0 hours, 0 minutes, 10 seconds
Next send starts at:                  now
Replicated tags:
Data transfer compression:            Off
ssh strict host key checking:         On
Debug log level:                      3
Replication ID:                       0x4d7d34a
```

As the `grid` OS user on the standby GGhub node where ACFS is currently mounted, use the following command to monitor the status of the ACFS replication:

```
[grid@gghub_stby1 ~]$ /sbin/acfsutil repl info -c -v /mnt/acfs_gg1

Site:                                 Standby
Primary hostname:                     gghub_prim_vip1.frankfurt.goldengate.com
Primary path:                         /mnt/acfs_gg1

Standby connect string:
grid@gghub_stby_vip1.frankfurt.goldengate.com
Standby path:                         /mnt/acfs_gg1
Replication interval:                 0 days, 0 hours, 0 minutes, 0 seconds
Last sync time with primary:          Fri May 05 12:37:02 2023
Receiving primary as of:              Fri May 05 12:37:02 2023
Status:                               Receive Completed
Last receive started at:              Fri May 05 12:37:02 2023
Last receive completed at:            Fri May 05 12:37:07 2023
Elapsed time for last receive:        0 days, 0 hours, 0 minutes, 5 seconds
Data transfer compression:            Off
ssh strict host key checking:         On
Debug log level:                      3
Replication ID:                       0x4d7d34a
```

Step 3.3.8 – Create the ACFS Replication CRS Action Scripts

To determine the health of the ACFS primary and standby file systems, CRS action scripts are used. At predefined intervals the action scripts report the health of the file systems into the CRS trace file `crsd_scriptagent_grid.trc` (or `crsd_scriptagent_oracle.trc` if role separation is not used) located in the Grid Infrastructure trace file directory `/u01/app/grid/diag/crs/`*`node_name`*`/crs/trace` on each of the primary and standby file system of the GGhub nodes.

On both the primary and standby file system clusters, there are two scripts required. One to monitor the local primary file system, and if the remote standby file system is available, and one to monitor the local standby file system and check remote primary file systems' availability. Example scripts are provided to implement the ACFS monitoring, but you must edit them to suit your environment.

Each replicated file system will need its own `acfs_primary` and `acfs_standby` action scripts.

Step 3.3.8.1 - Action Script acfs_primary.scr

The `acfs_primary` CRS resource checks whether the current ACFS mount is a primary file system and confirms that the standby file system is accessible and receiving replicated data. The resource is used to automatically determine if Oracle GoldenGate can start processes on the primary Oracle GoldenGate hub. If the standby file system is not accessible by the primary, the example script makes multiple attempts to verify the standby file system.

The `acfs_primary` CRS resource runs on both, the primary and standby hosts, but only returns success when the current file system is the primary file system, and the standby file system is accessible. The script must be placed in the same location on all primary and standby file system nodes.

The following parameters use suggested default settings, which should be tested before changing their values:

- `MOUNT_POINT=/mnt/acfs_gg1`
  `# The replicated ACFS mount point`
- `PATH_NAME=$MOUNT_POINT/status/acfs_primary`
  `# Must be unique from other mount files`
- `ATTEMPTS=3`
  `# Number of attempts to check the remote standby file system`
- `INTERVAL=10`
  `# Number of seconds between each attempt`

As the `grid` OS user on all primary and standby GGHub nodes, edit the `acfs_primary.scr` script to match the environment:

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ vi /u01/oracle/scripts/acfs_primary.scr
```

As the `oracle` OS user on the primary GGhub node where ACFS is currently mounted, run the following commands to create the `status` directory:

```
[opc@gghub_prim1 ~]$ sudo su - oracle
[oracle@gghub_prim1 ~]$ mkdir /mnt/acfs_gg1/status
[oracle@gghub_prim1 ~]$ chmod g+w /mnt/acfs_gg1/status
```

As the `grid` OS user on the primary and standby GGHub node where ACFS is currently mounted, run the following command to register the `acfs_primary` action script for monitoring the primary and standby file system:

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ sh /u01/oracle/scripts/add_acfs_primary.sh

##############################################################################
##
List of ACFS resources:
ora.data.acfs_gg1.acfs
##############################################################################
##
ACFS resource name: <ora.data.acfs_gg1.acfs>
```

As the `grid` OS user on the primary GGhub node where ACFS is currently mounted, start and check the status of the `acfs_primary` resource:

```
[grid@gghub_prim1 ~]$ crsctl start resource acfs_primary

CRS-2672: Attempting to start 'acfs_primary' on 'gghub_prim1'
CRS-2676: Start of 'acfs_primary' on 'gghub_prim1' succeeded


[grid@gghub_prim1 ~]$ crsctl stat resource acfs_primary

NAME=acfs_primary
TYPE=cluster_resource
TARGET=ONLINE
STATE=ONLINE on gghub_prim1
```

```
[grid@gghub_prim1 ~]$
 grep acfs_primary /u01/app/grid/diag/crs/`hostname`/crs/trace/
crsd_scriptagent_grid.trc
 |grep check

2023-05-05 12:57:40.372 :CLSDYNAM:2725328640: [acfs_primary]{1:33562:34377}
[check]
 Executing action script: /u01/oracle/scripts/acfs_primary.scr[check]
2023-05-05 12:57:42.376 :CLSDYNAM:2725328640: [acfs_primary]{1:33562:34377}
[check]
 SUCCESS: STANDBY file system /mnt/acfs_gg1 is ONLINE
```

As the `grid` OS user on the standby GGhub node where ACFS is currently mounted, start and check the status of the `acfs_primary` resource.

This step should fail because `acfs_primary` should **ONLY** be online on the primary GGhub:

```
[grid@gghub_stby1 ~]$ crsctl start res acfs_primary -n `hostname`

CRS-2672: Attempting to start 'acfs_primary' on 'gghub_stby1'
CRS-2674: Start of 'acfs_primary' on 'gghub_stby1' succeeded
CRS-2679: Attempting to clean 'acfs_primary' on 'gghub_stby1'
CRS-2681: Clean of 'acfs_primary' on 'gghub_stby1' succeeded
CRS-4000: Command Start failed, or completed with errors.


[grid@gghub_stby1 ~]$ crsctl stat res acfs_primary

NAME=acfs_primary
TYPE=cluster_resource
TARGET=ONLINE
STATE=OFFLINE

[grid@gghub_stby1 trace]$ grep
 acfs_primary /u01/app/grid/diag/crs/`hostname`/crs/trace/
crsd_scriptagent_grid.trc
 |grep check

2023-05-05 13:09:53.343 :CLSDYNAM:3598239488: [acfs_primary]{1:8532:2106}
[check]
 Executing action script: /u01/oracle/scripts/acfs_primary.scr[check]
2023-05-05 13:09:53.394 :CLSDYNAM:3598239488: [acfs_primary]{1:8532:2106}
[check]
 Detected local standby file system
2023-05-05 13:09:53.493 :CLSDYNAM:1626130176: [acfs_primary]{1:8532:2106}
[clean]
 Clean/Abort -- Stopping ACFS file system type checking...
```

> **Note:**
>
> The status of the `acfs_primary` resources will only be `ONLINE` if the ACFS file system is the primary file system. When starting the resources on a node which is not currently on the primary cluster, an error is reported because the resource fails due to being the standby file system. This error can be ignored. The resource will be in `OFFLINE` status on the ACFS standby cluster.

Step 3.3.8.2 - Action Script acfs_standby.scr

The `acfs_standby` resource checks that the local file system is a standby file system and verifies the remote primary file system status. If the primary file system fails verification multiple times (controlled by the action script variables), a warning is output to the CRS trace file `crsd_scriptagent_grid.trc` (or `crsd_scriptagent_oracle.trc` if role separation is not used) located in the Grid Infrastructure trace file directory `/u01/app/grid/diag/crs/node_name/crs/trace`.

This resource runs on both the primary and standby hosts, but only returns success when the current file system is the standby file system, and the primary file system is accessible.

The following parameters use suggested default settings, which should be tested before changing their values.

- `MOUNT_POINT=/mnt/acfs_gg1`
  `# This is the replicated ACFS mount point`
- `ATTEMPTS=3`
  `# Number of tries to check the remote primary file system`
- `INTERVAL=10`
  `# Number of seconds between each attempt`

As the `grid` OS user on all primary and standby GGHub nodes, edit the `acfs_standby.scr` script to match the environment:

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ vi /u01/oracle/scripts/acfs_standby.scr
```

As the `grid` OS user on the primary and standby GGHub node where ACFS is currently mounted, run the following command to register the `acfs_standby` action script for monitoring the primary and standby file system:

```
[grid@gghub_prim1 ~]$ sh /u01/oracle/scripts/add_acfs_standby.sh

################################################################################
##
List of VIP resources:
gghub_prim1_vip1
gghub_prim1_vip2
################################################################################
##
Application VIP CRS Resource: <gghub_prim1_vip1>
################################################################################
##
List of ACFS resources:
ora.data.acfs_gg1.acfs
```

```
################################################################################
##
ACFS resource name: <ora.data.acfs_gg1.acfs>
```

As the `grid` OS user on the primary and standby GGHub node where ACFS is currently mounted, start and check the status of the `acfs_standby` resource:

```
[grid@gghub_prim1 ~]$ crsctl start res acfs_standby

CRS-2672: Attempting to start 'acfs_standby' on 'gghub_prim1'
CRS-2676: Start of 'acfs_standby' on 'gghub_prim1' succeeded

[grid@gghub_prim1 ~]$ grep acfs_standby
 /u01/app/grid/diag/crs/`hostname`/crs/trace/crsd_scriptagent_grid.trc |egrep
'check|INFO'

2023-05-05 13:22:09.612 :CLSDYNAM:2725328640: [acfs_standby]{1:33562:34709}
[start]
 acfs_standby.scr starting to check ACFS remote primary at /mnt/acfs_gg1
2023-05-05 13:22:09.612 :CLSDYNAM:2725328640: [acfs_standby]{1:33562:34709}
[check]
 Executing action script: /u01/oracle/scripts/acfs_standby.scr[check]
2023-05-05 13:22:09.663 :CLSDYNAM:2725328640: [acfs_standby]{1:33562:34709}
[check]
 Local PRIMARY file system /mnt/acfs_gg1
```

Step 3.3.9 – Test ACFS GGhub Node Relocation

It is very important to test planned and unplanned ACFS GGhub node relocations and server role transitions before configuring Oracle GoldenGate.

As the `grid` OS user on the primary and standby GGHub nodes, copy the scripts from node 1 to node 2:

```
[grid@gghub_prim1 ~]$ scp -rq /u01/oracle/scripts gghub_prim2:/u01/oracle

[grid@gghub_stby1 ~]$ scp -rq /u01/oracle/scripts gghub_stby2:/u01/oracle
```

As the `grid` OS user on the primary and standby GGHub nodes, verify that the file system is mounted on another node, along with the VIP, `sshd_restart`, and the two ACFS resources (`acfs_primary` and `acfs_standby`) using the following example command:

```
[grid@gghub_prim1 ~]$ crsctl stat res sshd_restart acfs_primary
 acfs_standby ora.data.acfs_gg1.acfs sshd_restart -t

--------------------------------------------------------------------------------
--
Name           Target  State        Server                   State
details
--------------------------------------------------------------------------------
--
Cluster Resources
--------------------------------------------------------------------------------
--
```

```
acfs_primary
      1        ONLINE  ONLINE       gghub_prim2               STABLE
acfs_standby
      1        ONLINE  ONLINE                                 STABLE
gghubfad2
      1        ONLINE  ONLINE       gghub_prim2               STABLE
ora.data.acfs_gg1.acfs
      1        ONLINE  ONLINE       gghub_prim2               mounted on /mnt/
acfs
                                                              _gg1,STABLE
sshd_restart
      1        ONLINE  ONLINE       gghub_prim2               STABLE
--------------------------------------------------------------------------------
--

[grid@gghub_stby1 ~]$ crsctl stat res sshd_restart acfs_primary acfs_standby
 ora.data.acfs_gg1.acfs sshd_restart -t

--------------------------------------------------------------------------------
--
Name           Target  State        Server                   State details
--------------------------------------------------------------------------------
--
Cluster Resources
--------------------------------------------------------------------------------
--
acfs_primary
      1        ONLINE  OFFLINE                                STABLE
acfs_standby
      1        ONLINE  ONLINE       gghub_stby2               STABLE
ora.data.acfs_gg1.acfs
      1        ONLINE  ONLINE       gghub_stby2               mounted on /mnt/
acfs
                                                              _gg1,STABLE
sshd_restart
      1        ONLINE  ONLINE       gghub_stby2               STABLE
--------------------------------------------------------------------------------
--
```

Step 3.3.10 – Test ACFS Switchover Between the Primary and Standby GGhub

As the `grid` OS user on the standby GGHub node, run the following command to issue an ACFS switchover (role reversal) between the primary and standby GGhub:

```
[grid@gghub_stby2 ~]$ crsctl stat res ora.data.acfs_gg1.acfs

NAME=ora.data.acfs_gg.acfs
TYPE=ora.acfs_cluster.type
TARGET=ONLINE
STATE=ONLINE on gghub_stby2

[grid@gghub_stby2 ~]$ acfsutil repl failover /mnt/acfs_gg1

[grid@gghub_stby2 ~]$ /sbin/acfsutil repl info -c -v /mnt/acfs_gg1

Site:                                 Primary
```

```
Primary hostname:                       gghub_stby_vip1.frankfurt.goldengate.com
Primary path:                           /mnt/acfs_gg1
Primary status:                         Running
Background Resources:                   Active

Standby connect string:                 gghub_prim_vip1.frankfurt.goldengate.com
Standby path:                           /mnt/acfs_gg1
Replication interval:                   0 days, 0 hours, 0 minutes, 0 seconds
Sending primary as of:                  Fri May 05 13:51:37 2023
Status:                                 Send Completed
Lag Time:                               00:00:00
Retries made:                           0
Last send started at:                   Fri May 05 13:51:37 2023
Last send completed at:                 Fri May 05 13:51:48 2023
Elapsed time for last send:             0 days, 0 hours, 0 minutes, 11 seconds
Next send starts at:                    now
Replicated tags:
Data transfer compression:              Off
ssh strict host key checking:           On
Debug log level:                        3
Replication ID:                         0x4d7d34a
```

As the `grid` OS user on the new standby GGHub node (old primary), run the following command to issue an ACFS switchover (role reversal) between the primary and standby GGhub.

This step is optional but recommended to return the sites to the original role:

```
[grid@gghub_prim2 ~]$ crsctl stat res ora.data.acfs_gg1.acfs

NAME=ora.data.acfs_gg1.acfs
TYPE=ora.acfs_cluster.type
TARGET=ONLINE
STATE=ONLINE on gghub_prim2

[grid@gghub_prim2 ~]$ /sbin/acfsutil repl info -c -v /mnt/acfs_gg1 |grep Site
Site:                                   Standby

[grid@gghub_prim2 ~]$ acfsutil repl failover /mnt/acfs_gg1

[grid@gghub_prim2 ~]$ /sbin/acfsutil repl info -c -v /mnt/acfs_gg1 |grep Site

Site:                                   Primary
```

Step 3.4 - Create the Oracle GoldenGate Deployment

Once the Oracle GoldenGate software has been installed in the GGHub, the next step is to create a response file to create the GoldenGate deployment using the Oracle GoldenGate Configuration Assistant.

The unified build feature introduced in Oracle GoldenGate 21c means a single deployment can now manage Extract and Replicat processes that attach to different Oracle Database versions. Each deployment is created with an Administration Server and (optionally) Performance Metrics Server. If the GoldenGate trail files don't need to be transferred to another hub or GoldenGate environment, there is no need to create a Distribution or Receiver Server.

Two limitations currently exist with Oracle GoldenGate and XAG:

1. A Service Manager that is registered with XAG can only manage a single deployment. If multiple deployments are required, each deployment must use its own Service Manager. Oracle GoldenGate release 21c simplifies this requirement because it uses a single deployment to support Extract and Relicat processes connecting to different versions of the Oracle Database.
2. Each Service Manager registered with XAG must belong to separate OGG_HOME software installation directories. Instead of installing Oracle GoldenGate multiple times, the recommended approach is to install Oracle GoldenGate one time, and then create a symbolic link for each Service Manager OGG_HOME. The symbolic link and OGG_HOME environment variable must be configured **before** running the Oracle GoldenGate Configuration Assistant on all Oracle RAC nodes.

Create a Response File

For a silent configuration, copy the following example file and paste it into any location the oracle user can access. Edit the following values appropriately:

- `CONFIGURATION_OPTION`
- `DEPLOYMENT_NAME`
- `ADMINISTRATOR_USER`
- `SERVICEMANAGER_DEPLOYMENT_HOME`
- `OGG_SOFTWARE_HOME`
- `OGG_DEPLOYMENT_HOME`
- `ENV_TNS_ADMIN`
- `OGG_SCHEMA`

Example Response File (`oggca.rsp`):

As the `oracle` OS user on the primary GGHub node where ACFS is currently mounted, create and edit the response file `oggca.rsp` to create the Oracle GoldenGate deployment:

```
[opc@gghub_prim1 ~]$ sudo su - oracle
[oracle@gghub_prim1 ~]$ vi /u01/oracle/scripts/oggca.rsp

oracle.install.responseFileVersion=/oracle/install/
rspfmt_oggca_response_schema_v21_1_0
CONFIGURATION_OPTION=ADD
DEPLOYMENT_NAME=<GG_DEPLOYMENT_NAME>
ADMINISTRATOR_USER=oggadmin
ADMINISTRATOR_PASSWORD=<password_for_oggadmin>
SERVICEMANAGER_DEPLOYMENT_HOME=/mnt/acfs_gg1/deployments/ggsm01
HOST_SERVICEMANAGER=localhost
PORT_SERVICEMANAGER=9100
SECURITY_ENABLED=false
STRONG_PWD_POLICY_ENABLED=true
CREATE_NEW_SERVICEMANAGER=true
REGISTER_SERVICEMANAGER_AS_A_SERVICE=false
INTEGRATE_SERVICEMANAGER_WITH_XAG=true
EXISTING_SERVICEMANAGER_IS_XAG_ENABLED=false
OGG_SOFTWARE_HOME=/u01/app/oracle/goldengate/gg21c
OGG_DEPLOYMENT_HOME=/mnt/acfs_gg1/deployments/gg01
ENV_LD_LIBRARY_PATH=${OGG_HOME}/lib/instantclient:${OGG_HOME}/lib
```

```
ENV_TNS_ADMIN=/u01/app/oracle/goldengate/network/admin
FIPS_ENABLED=false
SHARDING_ENABLED=false
ADMINISTRATION_SERVER_ENABLED=true
PORT_ADMINSRVR=9101
DISTRIBUTION_SERVER_ENABLED=true
PORT_DISTSRVR=9102
NON_SECURE_DISTSRVR_CONNECTS_TO_SECURE_RCVRSRVR=false
RECEIVER_SERVER_ENABLED=true
PORT_RCVRSRVR=9103
METRICS_SERVER_ENABLED=true
METRICS_SERVER_IS_CRITICAL=false
PORT_PMSRVR=9104
UDP_PORT_PMSRVR=9105
PMSRVR_DATASTORE_TYPE=BDB
PMSRVR_DATASTORE_HOME=/u01/app/oracle/goldengate/datastores/
<GG_DEPLOYMENT_NAME>
OGG_SCHEMA=ggadmin
```

Create the Oracle GoldenGate Deployment

As the `oracle` OS user on the primary GGHub node where ACFS is currently mounted, run `oggca.sh` to create the GoldenGate deployment:

```
[opc@gghub_prim1 ~]$ sudo su - oracle
[oracle@gghub_prim1 ~]$ export OGG_HOME=/u01/app/oracle/goldengate/gg21c
[oracle@gghub_prim1 ~]$ $OGG_HOME/bin/oggca.sh -silent
 -responseFile /u01/oracle/scripts/oggca.rsp

Successfully Setup Software.
```

Create the Oracle GoldenGate Datastores and TNS_ADMIN Directories

As the `oracle` OS user on all GGHub nodes of the primary and standby systems, run the following commands to create the Oracle GoldenGate Datastores and `TNS_ADMIN` directories:

```
[opc@gghub_prim1 ~]$ sudo su - oracle
[oracle@gghub_prim1 ~]$ mkdir -p /u01/app/oracle/goldengate/network/admin
[oracle@gghub_prim1 ~]$ mkdir -p /u01/app/oracle/goldengate/datastores/
<GG_DEPLOYMENT_NAME>
```

Step 3.5 - Configure Oracle Grid Infrastructure Agent (XAG)

The following step-by-step procedure shows you how to configure Oracle Clusterware to manage GoldenGate using the Oracle Grid Infrastructure Standalone Agent (XAG). Using XAG automates the ACFS file system mounting, as well as the stopping and starting of the GoldenGate deployment when relocating between Oracle GGhub nodes.

Step 3.5.1 - Install the Oracle Grid Infrastructure Standalone Agent

It is recommended that you install the XAG software as a standalone agent outside the Grid Infrastructure `ORACLE_HOME` so that you can use the latest XAG release available, and the software can be updated without impact to the Grid Infrastructure.

Install the XAG standalone agent outside of the Oracle Grid Infrastructure home directory. XAG must be installed in the same directory on all GGhub nodes in the system where GoldenGate is installed.

As the `grid` OS user on the first GGHub node of the primary and standby systems, unzip the software and run `xagsetup.sh`:

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ unzip /u01/oracle/stage/p31215432_190000_Generic.zip
 -d /u01/oracle/stage
[grid@gghub_prim1 ~]$ /u01/oracle/stage/xag/xagsetup.sh --install
 --directory /u01/app/grid/xag --all_nodes

Installing Oracle Grid Infrastructure Agents on: gghub_prim1
Installing Oracle Grid Infrastructure Agents on: gghub_prim2
Updating XAG resources.
Successfully updated XAG resources.
```

As the `grid` OS user on all GGHub nodes of the primary and standby systems, add the location of the newly installed XAG software to the `PATH` variable so that the location of `agctl` is known when the `grid` user logs on to the machine.

```
[grid@gghub_prim1 ~]$ vi ~/.bashrc

PATH=/u01/app/grid/xag/bin:$PATH:/u01/app/19.0.0.0/grid/bin; export PATH
```

> **Note:**
>
> It is essential to ensure that the XAG bin directory is specified BEFORE the Grid Infrastructure bin directory to ensure the correct `agctl` binary is found. This should be set in the grid user environment to take effect when logging on, such as in the .bashrc file when the Bash shell is in use.

Step 3.5.2 - Register Oracle Grid Infrastructure Agent on the Primary and Standby GGHubs

The following procedure shows you how to configure Oracle Clusterware to manage Oracle GoldenGate using the Oracle Grid Infrastructure Standalone Agent (XAG). Using XAG automates the mounting of the shared file system as well as the stopping and starting of the Oracle GoldenGate deployment when relocating between Oracle GGhub nodes.

Oracle GoldenGate must be registered with XAG so that the deployment is started and stopped automatically when the database is started, and the file system is mounted.

To register Oracle GoldenGate Microservices Architecture with XAG, use the following command format.

```
agctl add goldengate <instance_name>
--gg_home <GoldenGate_Home>
--service_manager
--config_home <GoldenGate_SvcMgr_Config>
--var_home <GoldenGate_SvcMgr_Var Dir>
--oracle_home <$OGG_HOME/lib/instantclient>
--port <port number>
```

```
--adminuser <OGG admin user>
--user <GG instance user>
--group <GG instance group>
--file systems <CRS_resource_name>
--filesystems_always yes
--filesystem_verify <yes/no>
--attribute TARGET_DEFAULT=online
```

Where:

- `--gg_home` specifies the location of the GoldenGate software.
- `--service_manager` indicates this is an GoldenGate Microservices instance.
- `--config_home` specifies the GoldenGate deployment configuration home directory.
- `--var_home` specifies the GoldenGate deployment variable home directory.
- `--oracle_home` specifies the Oracle Instant Client home
- `--port` specifies the deployment Service Manager port number.
- `--adminuser` specifies the GoldenGate Microservices administrator account name.
- `--user` specifies the name of the operating system user that owns the GoldenGate deployment.
- `--group` specifies the name of the operating system group that owns the GoldenGate deployment.
- `--filesystems` specifies the CRS file system resource that must be ONLINE before the deployment is started. This will be the `acfs_primary` resource created in a previous step.
- `--filesystem_verify` specifies if XAG should check the existence of the directories specified by the `config_home` and `var_home` parameters. This should be set to 'yes' for the active ACFS primary file system. When adding the GoldenGate instance on the standby cluster, specify 'no'.
- `--filesystems_always` specifies that XAG will start the GoldenGate Service Manager on the same GGhub node as the file system CRS resources, specified by the `--filesystems` parameter.
- `--attributes` specifies that the target status of the resource is online. This is required to automatically start the GoldenGate deployment when the `acfs_primary` resource starts.

The GoldenGate deployment must be registered on the primary and standby GGHubs where ACFS is mounted in either read-write or read-only mode.

As the `grid` OS user on the first GGHub node of the primary and standby systems, run the following command to determine which node of the cluster the file system is mounted on:

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ crsctl stat res acfs_standby |grep STATE
STATE=ONLINE on gghub_prim1
```

Step 3.5.2.1 - Register the Primary Oracle GoldenGate Microservices Architecture with XAG

As the `root` OS user on the first node of the primary GGHub, register Oracle GoldenGate Microservices Architecture with XAG using the following command format:

```
[opc@gghub_prim1 ~]$ sudo su - root
[root@gghub_prim1 ~]# grep DEPLOYMENT_NAME= /u01/oracle/scripts/oggca.rsp
DEPLOYMENT_NAME=<gghub1>
[root@gghub_prim1 ~]# export GG_DEPLOYMENT_NAME=<gghub1>

[root@gghub_prim1 ~]# vi /u01/oracle/scripts/add_xag_goldengate_prim.sh

# Run as ROOT:

/u01/app/grid/xag/bin/agctl add goldengate $GG_DEPLOYMENT_NAME \
--gg_home /u01/app/oracle/goldengate/gg21c \
--service_manager \
--config_home /mnt/acfs_gg1/deployments/ggsm01/etc/conf \
--var_home /mnt/acfs_gg1/deployments/ggsm01/var \
--oracle_home /u01/app/oracle/goldengate/gg21c/lib/instantclient \
--port 9100 \
--adminuser oggadmin \
--user oracle \
--group oinstall \
--filesystems acfs_primary \
--filesystems_always yes \
--filesystem_verify yes \
--attribute TARGET_DEFAULT=online

[root@gghub_prim1 ~]# sh /u01/oracle/scripts/add_xag_goldengate_prim.sh
Enter password for 'oggadmin' : ##########
```

As the `grid` OS user on the first node of the primary GGHub, verify that Oracle GoldenGate Microservices Architecture is registered with XAG:

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ agctl status goldengate

Goldengate  instance 'gghub1' is not running
```

As the `grid` OS user on the first node of the primary GGHub, add the environment variable `GG_DEPLOYMENT_NAME` to the `~/.bashrc` file:

```
[grid@gghub_prim1 ~]$ cat >> ~/.bashrc <<EOF
export GG_DEPLOYMENT_NAME=`/u01/app/grid/xag/bin/agctl status goldengate |
 awk '{print $3}' | tr -d "'"`
EOF

[grid@gghub_prim1 ~]$ . ~/.bashrc
[grid@gghub_prim1 ~]$ echo $GG_DEPLOYMENT_NAME

gghub1
```

Step 3.5.2.2 - Register the Standby Oracle GoldenGate Microservices Architecture with XAG

As the `root` OS user on the first node of the standby GGHub, register Oracle GoldenGate Microservices Architecture with XAG using the following command format:

```
[opc@gghub_stby1 ~]$ sudo su - root
[root@gghub_stby1 ~]# vi /u01/oracle/scripts/add_xag_goldengate_stby.sh
[root@gghub_stby1 ~]# export GG_DEPLOYMENT_NAME=<gghub1>

# Run as ROOT:

/u01/app/grid/xag/bin/agctl add goldengate $GG_DEPLOYMENT_NAME \
--gg_home /u01/app/oracle/goldengate/gg21c \
--service_manager \
--config_home /mnt/acfs_gg1/deployments/ggsm01/etc/conf \
--var_home /mnt/acfs_gg1/deployments/ggsm01/var \
--oracle_home /u01/app/oracle/goldengate/gg21c/lib/instantclient \
--port 9100 --adminuser oggadmin --user oracle --group oinstall \
--filesystems acfs_primary \
--filesystems_always yes \
--filesystem_verify no \
--attribute TARGET_DEFAULT=online

[root@gghub_stby1 ~]# sh /u01/oracle/scripts/add_xag_goldengate_stby.sh
Enter password for 'oggadmin' : ##########
```

> **Note:**
>
> When adding the GoldenGate instance on the standby cluster, specify `--filesystem_verify no`.

As the `grid` OS user on the first node of the standby GGHub, verify that Oracle GoldenGate Microservices Architecture is registered with XAG:

```
[opc@gghub_stby1 ~]$ sudo su - grid
[grid@gghub_stby1 ~]$ agctl status goldengate

Goldengate  instance 'gghub1' is not running
```

As the `grid` OS user on the first node of the standby GGHub, add the environment variable `GG_DEPLOYMENT_NAME` to the ~/.bashrc file:

```
[grid@gghub_stby1 ~]$ cat >> ~/.bashrc <<EOF
export GG_DEPLOYMENT_NAME=`/u01/app/grid/xag/bin/agctl status goldengate |
 awk '{print $3}' | tr -d "'"`
EOF

[grid@gghub_stby1 ~]$ . ~/.bashrc
[grid@gghub_prim1 ~]$ echo $GG_DEPLOYMENT_NAME

gghub1
```

Step 3.5.3 - Start the Oracle GoldenGate Deployment

Below are some example `agctl` commands used to manage the GoldenGate deployment with XAG.

As the `grid` OS user on the first node of the primary GGHub, execute the following command to start and check Oracle GoldenGate deployment:

```
[opc@gghub_prim1 ~]$ sudo su - grid

[grid@gghub_prim1 ~]$ agctl start goldengate $GG_DEPLOYMENT_NAME

[grid@gghub_prim1 ~]$ agctl status goldengate
Goldengate  instance 'gghub1' is running on gghub_prim1
```

As the `grid` OS user on the first GGHub node, run the following command to validate the configuration parameters for the Oracle GoldenGate resource:

```
[grid@gghub_prim1 ~]$ agctl config goldengate $GG_DEPLOYMENT_NAME

Instance name: gghub1
Application GoldenGate location is: /u01/app/oracle/goldengate/gg21c
Goldengate MicroServices Architecture environment: yes
Goldengate Service Manager configuration directory: /mnt/acfs_gg1/deployments/
ggsm01/etc/conf
Goldengate Service Manager var directory: /mnt/acfs_gg1/deployments/ggsm01/var
Service Manager Port: 9100
Goldengate Administration User: oggadmin
Autostart on DataGuard role transition to PRIMARY: no
ORACLE_HOME location is: /u01/app/oracle/goldengate/gg21c/lib/instantclient
File System resources needed: acfs_primary
CRS additional attributes set: TARGET_DEFAULT=online
```

For more information see Oracle Grid Infrastructure Bundled Agent.

Step 3.6 - Configure NGINX Reverse Proxy

The Oracle GoldenGate reverse proxy feature allows a single point of contact for all the GoldenGate microservices associated with a GoldenGate deployment. Without a reverse proxy, the GoldenGate deployment microservices are contacted using a URL consisting of a hostname or IP address and separate port numbers, one for each of the services. For example, to contact the Service Manager, you could use http://gghub.example.com:9100, then the Administration Server is http://gghub.example.com:9101, the second Service Manager may be accessed using http://gghub.example.com:9110, and so on.

When running Oracle GoldenGate in a High Availability (HA) configuration on Oracle Exadata Database Service with the Grid Infrastructure agent (XAG), there is a limitation preventing more than one deployment from being managed by a GoldenGate Service Manager. Because of this limitation, creating a separate virtual IP address (VIP) for each Service Manager/ deployment pair is recommended. This way, the microservices can be accessed directly using the VIP.

With a reverse proxy, port numbers are not required to connect to the microservices because they are replaced with the deployment name and the hostname's VIP. For example, to connect to the console via a web browser, use the URLs:

GoldenGate Services	URL
Service Manager	https://localhost:443
Administration Server	https://localhost:443/*instance_name*/adminsrvr
Distribution Server	https://localhost:443/*instance_name*/distsrvr
Performance Metric Server	https://localhost:443/*instance_name*/pmsrvr
Receiver Server	https://localhost:443/*instance_name*/recvsrvr

When running multiple Service Managers, the following instructions will provide configuration using a separate VIP for each Service Manager. NGINX uses the VIP to determine which Service Manager an HTTPS connection request is routed to.

An SSL certificate is required for clients to authenticate the server they connect to through NGINX. Contact your systems administrator to follow your corporate standards to create or obtain the server certificate before proceeding. A separate certificate is required for each VIP and Service Manager pair.

> **Note:**
>
> The common name in the CA-signed certificate must match the target hostname/VIP used by NGINX.

Follow the instructions to install and configure NGINX Reverse Proxy with an SSL connection and ensure all external communication is secure.

Step 3.6.1 - Secure Deployments Requirements (Certificates)

A secure deployment involves making RESTful API calls and conveying trail data between the Distribution Server and Receiver Server, over SSL/TLS.

You can use your own existing business certificate from your Certificate Authority (CA) or you might create your own certificates.

Contact your systems administrator to follow your corporate standards to create or obtain the server certificate before proceeding. A separate certificate is required for each VIP and Service Manager pair.

Step 3.6.2 - Install NGINX Reverse Proxy Server

As the `root` OS user on all GGHub nodes, set up the `yum` repository by creating the file `/etc/yum.repos.d/nginx.repo` with the following contents:

```
[opc@gghub_prim1 ~]$ sudo su -
[root@gghub_prim1 ~]# cat > /etc/yum.repos.d/nginx.repo <<EOF
[nginx-stable]
name=nginx stable repo
baseurl=http://nginx.org/packages/rhel/7/\$basearch/
gpgcheck=1
enabled=1
gpgkey=https://nginx.org/keys/nginx_signing.key
module_hotfixes=true
EOF
```

As the `root` OS user on all GGHub nodes, run the following commands to install, enable, and start NGINX:

```
[root@gghub_prim1 ~]# yum install -y python-requests python-urllib3 nginx
[root@gghub_prim1 ~]# systemctl enable nginx
```

As the `root` OS user on all GGHub node, disable the NGINX repository after the software has been installed:

```
[root@gghub_prim1 ~]# yum-config-manager --disable nginx-stable
```

Step 3.6.3 - Create the NGINX Configuration File

You can configure Oracle GoldenGate Microservices Architecture to use a reverse proxy. Oracle GoldenGate MA includes a script called `ReverseProxySettings` that generates a configuration file for only the NGINX reverse proxy server.

The script requires the following parameters:

- The --user parameter should mirror the GoldenGate administrator account specified with the initial deployment creation.
- The GoldenGate administrator password will be prompted.
- The reverse proxy port number specified by the --port parameter should be the default HTTPS port number (443) unless you are running multiple GoldenGate Service Managers using the same --host. In this case, specify an HTTPS port number that does not conflict with previous Service Manager reverse proxy configurations. For example, if running two Service Managers using the same hostname/VIP, the first reverse proxy configuration is created with '--port 443 --host VIP_NAME1.FQDN', and the second is created with '--port 444 --host VIP_NAME2.FQDN'. If using separate hostnames/VIPs, the two Service Manager reverse proxy configurations would be created with '--port 443 --host VIP_NAME1.FQDN' and '--port 443 --host VIP_NAME2.FQDN'.
- The --host parameter is the VIP_NAME.FQDN configured in the Private DNS Zone View
- Lastly, the HTTP port number (9100) should match the Service Manager port number specified during the deployment creation.

Repeat this step for each additional GoldenGate Service Manager.

As the `oracle` OS user on the first GGHub node, use the following command to create the Oracle GoldenGate NGINX configuration file:

```
[oracle@gghub_prim1 ~]$ export OGG_HOME=/u01/app/oracle/goldengate/gg21c
[oracle@gghub_prim1 ~]$ export PATH=$PATH:$OGG_HOME/bin
[oracle@gghub_prim1 ~]$ cd /u01/oracle/scripts
[oracle@gghub_prim1 ~]$ $OGG_HOME/lib/utl/reverseproxy/ReverseProxySettings
 --user oggadmin --port 443 --output ogg_$GG_DEPLOYMENT_NAME.conf http://
localhost:9100
 --host <VIP_NAME.FQDN>
Password: <oggadmin_password>
```

Step 3.6.4 - Modify NGINX Configuration Files

When multiple GoldenGate Service Managers are configured to use their IP/VIPs with the same HTTPS 443 port, some small changes are required to the NGINX reverse proxy configuration files generated in the previous step. With all Service Managers sharing the same

port number, they are independently accessed using their VIP/IP specified by the --host parameter.

As the `oracle` OS user on the first GGHub node, determine the deployment name managed by this Service Manager listed in the reverse proxy configuration file and change all occurrences of "_ServiceManager" by prepending the deployment name before the underscore:

```
[oracle@gghub_prim1 ~]$ cd /u01/oracle/scripts
[oracle@gghub_prim1 ~]$ grep "Upstream Servers" ogg_$GG_DEPLOYMENT_NAME.conf

## Upstream Servers for Deployment 'gghub1'

[oracle@gghub_prim1 ~]$ sed -i 's/_ServiceManager/
<REPLACE_WITH_DEPLOYMENT_NAME>_ServiceManager/' ogg_$GG_DEPLOYMENT_NAME.conf
```

Step 3.6.5 - Install the Server Certificates for NGINX

As the `root` OS user on the first GGHub node, copy the server certificates and key files in the `/etc/nginx/ssl` directory, owned by `root` with file permissions 400 (-r--------):

```
[opc@gghub_prim1 ~]$ sudo su -
[root@gghub_prim1 ~]# mkdir /etc/nginx/ssl
[root@gghub_prim1 ~]# cp <ssl_keys> /etc/nginx/ssl/.
[root@gghub_prim1 ~]# chmod -R 400 /etc/nginx/ssl
[root@gghub_prim1 ~]# ll /etc/nginx/ssl

-r-------- 1 root root 2750 May 17 06:12 gghub1.chained.crt
-r-------- 1 root root 1675 May 17 06:12 gghub1.key
```

As the `oracle` OS user on the first GGHub node, set the correct file names for the certificate and key files for each reverse proxy configuration file:

```
[oracle@gghub_prim1 ~]$ vi /u01/oracle/scripts/ogg_$GG_DEPLOYMENT_NAME.conf

# Before
    ssl_certificate         /etc/nginx/ogg.pem;
    ssl_certificate_key     /etc/nginx/ogg.pem;

# After
    ssl_certificate         /etc/nginx/ssl/gghub1.chained.crt;
    ssl_certificate_key     /etc/nginx/ssl/gghub1.key;
```

When using CA-signed certificates, the certificate named with the ssl_certificate NGINX parameter must include the 1) CA signed, 2) intermediate, and 3) root certificates in a single file. The order is significant; otherwise, NGINX fails to start and displays the error message:

```
(SSL: error:0B080074:x509 certificate routines:
  X509_check_private_key:key values mismatch)
```

The root and intermediate certificates can be downloaded from the CA-signed certificate provider.

As the `root` OS user on the first GGHub node, generate the SSL certificate single file by using the following example command:

```
[root@gghub_prim1 ~]# cd /etc/nginx/ssl
[root@gghub_prim1 ~]# cat CA_signed_cert.crt
 intermediate.crt root.crt > gghub1.chained.crt
```

The `ssl_certificate_key` file is generated when creating the Certificate Signing Request (CSR), which is required when requesting a CA-signed certificate.

Step 3.6.6 - Install the NGINX Configuration File

As the `root` OS user on the first GGhub node, copy the deployment configuration file to `/etc/nginx/conf.d` directory and remove the default configuration file:

```
[root@gghub_prim1 ~]# cp /u01/oracle/scripts/ogg_<gghub1>.conf
 /etc/nginx/conf.d
[root@gghub_prim1 ~]# rm /etc/nginx/conf.d/default.conf
```

As the `root` OS user on the first GGHub node, validate the NGINX configuration file. If there are errors in the file, they will be reported with the following command:

```
[root@gghub_prim1 ~]# nginx -t

nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginxconf test is successful
```

As the `root` OS user on the first GGHub node, restart NGINX to load the new configuration:

```
[root@gghub_prim1 ~]# systemctl restart nginx
```

Step 3.6.7 - Test GoldenGate Microservices Connectivity

As the `root` OS user on the first GGHub node, create a curl configuration file (`access.cfg`) that contains the deployment user name and password:

```
[root@gghub_prim1 ~]# vi access.cfg
user = "oggadmin:<password>"

[root@gghub_prim1 ~]# curl <--insecure> -svf -K access.cfg
 https://<vip_name.FQDN>:<port#>/services/v2/config/health -XGET && echo -e
 "\n*** Success"

Sample output:
* About to connect() to .frankfurt.goldengate.com port 443 (#0)
*   Trying 10.40.0.75...
* Connected to gghub_prim_vip1.frankfurt.goldengate.com (10.40.0.75) port 443
(#0)
* Initializing NSS with certpath: sql:/etc/pki/nssdb
*   CAfile: /etc/pki/tls/certs/ca-bundle.crt
  CApath: none
* skipping SSL peer certificate verification
* NSS: client certificate not found (nickname not specified)
* SSL connection using TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
```

```
* Server certificate:
*       subject: CN=gghub_prim_vip1.frankfurt.goldengate.com,OU=Oracle
 MAA,O=Oracle,L=Frankfurt,ST=Frankfurt,C=GE
*       start date: Jul 27 15:59:00 2023 GMT
*       expire date: Jul 26 15:59:00 2024 GMT
*       common name: gghub_prim_vip1.frankfurt.goldengate.com
*       issuer:
OID.2.5.29.19=CA:true,CN=gghub_prim_vip1.frankfurt.goldengate.com,OU=Oracle
MAA,O=Oracle,L=Frankfurt,C=EU
* Server auth using Basic with user 'oggadmin'
> GET /services/v2/config/health HTTP/1.1
> Authorization: Basic b2dnYWRtaW46V0VsY29tZTEyM19fXw==
> User-Agent: curl/7.29.0
> Host: gghub_prim_vip1.frankfurt.goldengate.com
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: nginx/1.24.0
< Date: Thu, 27 Jul 2023 16:25:26 GMT
< Content-Type: application/json
< Content-Length: 941
< Connection: keep-alive
< Set-Cookie:
ogg.sca.mS+pRfBERzqE+RTFZPPoVw=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOi
JvZ2cuc2NhIiwiZXhwIjozNjAwLCJ0eXAiOiJ4LVNDQS1BdXRob3JpemF0aW9uIiwic3ViIjoib2dn
YWRtaW4iLCJhdWQiOiJvZ2cuc2NhIiwiaWF0IjoxNjkwNDc1MTI2LCJob3N0IjoiZ2dodWJsYV92aX
AubG9uZG9uLmdvbGRlbmdhdGUuY29tIiwicm9sZSI6IlNlY3VyaXR5IiwiYXV0aFR5cGUiOiJCYXNp
YyIsImNyZWQiOiJFd3VqV0hOdzlGWDNHai9FN1RYU3A1N1dVRjBheUd4OFpCUTdiZDlKOU9RPSIsIn
NlcnZlcklEIjoiZmFkNWVkN2MtZThlYi00YmE2LTg4Y2EtNmQxYjk3ZjdiMGQ3IiwiZGVwbG95bWVu
dElEIjoiOTkyZmE5NDUtZjA0NC00NzNhLTg0ZjktMTRjNTY0ZjNlODU3In0=.knACABXPmZE4BEyux
7lZQ5GnrSCCh4x1zBVBLaX3Flo=; Domain=gghub_prim_vip1.frankfurt.goldengate.com;
Path=/; HttpOnly; Secure; SameSite=strict
< Set-Cookie:
ogg.csrf.mS+pRfBERzqE+RTFZPPoVw=1ae439e625798ee02f8f7498438f27c7bad036b270d6bf
c95aee60fcee111d35ea7e8dc5fb5d61a38d49cac51ca53ed9307f9cbe08fab812181cf163a743
bfc7; Domain=gghub_prim_vip1.frankfurt.goldengate.com; Path=/; Secure;
SameSite=strict
< Cache-Control: max-age=0, no-cache, no-store, must-revalidate
< Expires: 0
< Pragma: no-cache
< Content-Security-Policy: default-src 'self' 'unsafe-eval' 'unsafe-
inline';img-src 'self' data:;frame-ancestors https://
gghub_prim_vip1.frankfurt.goldengate.com;child-src https://
gghub_prim_vip1.frankfurt.goldengate.com blob:;
< X-Content-Type-Options: nosniff
< X-XSS-Protection: 1; mode=block
< X-OGG-Proxy-Version: v1
< Strict-Transport-Security: max-age=31536000 ; includeSubDomains
<
* Connection #0 to host gghub_prim_vip1.frankfurt.goldengate.com left intact
{"$schema":"api:standardResponse","links":[{"rel":"canonical","href":"https://
gghub_prim_vip1.frankfurt.goldengate.com/services/v2/config/
health","mediaType":"application/json"},{"rel":"self","href":"https://
gghub_prim_vip1.frankfurt.goldengate.com/services/v2/config/
health","mediaType":"application/json"},{"rel":"describedby","href":"https://
gghub_prim_vip1.frankfurt.goldengate.com/services/ServiceManager/v2/metadata-
```

```
catalog/health","mediaType":"application/schema+json"}],"messages":
[],"response":
{"$schema":"ogg:health","deploymentName":"ServiceManager","serviceName":"Servi
ceManager","started":"2023-07-27T15:39:41.867Z","healthy":true,"criticalResour
ces":
[{"deploymentName":"gghubl1","name":"adminsrvr","type":"service","status":"run
ning","healthy":true},
{"deploymentName":"gghubl","name":"distsrvr","type":"service","status":"runnin
g","healthy":true},
{"deploymentName":"gghubl","name":"recvsrvr","type":"service","status":"runnin
g","healthy":true}]}}
*** Success

[root@gghub_prim1 ~]# rm access.cfg
```

> **Note:**
>
> If the environment is using self-signed SSL certificates, add the flag --insecure to the curl command to avoid the error "NSS error -8172 (SEC_ERROR_UNTRUSTED_ISSUER)".

Step 3.6.8 - Remove NGINX default.conf Configuration File

As the `root` OS user on all GGHubs, remove the default configuration file (`default.conf`) created in `/etc/nginx/conf.d`:

```
[opc@gghub_prim1 ~]$ sudo rm -f /etc/nginx/conf.d/default.conf
[opc@gghub_prim1 ~]$ sudo nginx -s reload
```

Step 3.6.9 - Distribute the GoldenGate NGINX Configuration Files

Once all of the reverse proxy configuration files have been created for the GoldenGate Service Managers, they must be copied to the second GoldenGate Hub node.

As the `opc` OS user on the first GGHub node, distribute the NGINX configuration files to all database nodes:

```
[opc@gghub_prim1 ~]$ sudo tar fczP /tmp/nginx_conf.tar /etc/nginx/conf.d/
 /etc/nginx/ssl/
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ scp /tmp/nginx_conf.tar gghub_prim2:/tmp/.
```

As the `opc` OS user on the second GGHub node, extract the NGINX configuration files and remove the default configuration file:

```
[opc@gghub_prim2 ~]$ sudo tar fxzP /tmp/nginx_conf.tar
[opc@gghub_prim2 ~]$ sudo rm /etc/nginx/conf.d/default.conf
```

As the `opc` OS user on the second GGHub node, restart NGINX:

```
[opc@gghub_prim2 ~]$ sudo nginx -t
```

```
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful

[root@gghub_prim2 ~]$ sudo systemctl restart nginx
```

> **Note:**
>
> Repeat all of the steps in section 3.6 for the primary and standby GGHub systems.

Step 3.7 - Securing GoldenGate Microservices to Restrict Non-secure Direct Access

After configuring the NGINX reverse proxy with an unsecured Oracle GoldenGate Microservices deployment, the microservices can continue accessing HTTP (non-secure) using the configured microservices port numbers. For example, the following non-secure URL could be used to access the Administration Server: http://<vip-name>:9101.

Oracle GoldenGate Microservices' default behavior for each server (Service Manager, adminserver, pmsrvr. distsrvr, and recsrvr) is to listen using a configured port number on all network interfaces. This is undesirable for more secure installations, where direct access using HTTP to the Microservices needs to be disabled and only permitted using NGINX HTTPS.

Use the following commands to alter the Service Manager and deployment services listener address to use only the localhost address. Access to the Oracle GoldenGate Microservices will only be permitted from the localhost, and any access outside of the localhost will only succeed using the NGINX HTTPS port.

Step 3.7.1 - Stop the Service Manager

As the `grid` OS user on the first GGHub node, stop the GoldenGate deployment:

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ agctl stop goldengate $GG_DEPLOYMENT_NAME
[grid@gghub_prim1 ~]$ agctl status goldengate

Goldengate  instance 'gghub1' is not running
```

Step 3.7.2 - Modify the Service Manager Listener Address

As the `oracle` OS user on the first GGHub node, modify the listener address with the following commands. Use the correct port number for the Service Manager being altered:

```
[opc@gghub_prim1 ~]$ sudo su - oracle
[oracle@gghub_prim1 ~]$ export OGG_HOME=/u01/app/oracle/goldengate/gg21c
[oracle@gghub_prim1 ~]$ export OGG_VAR_HOME=/mnt/acfs_gg1/deployments/
ggsm01/var
[oracle@gghub_prim1 ~]$ export OGG_ETC_HOME=/mnt/acfs_gg1/deployments/
ggsm01/etc
[oracle@gghub_prim1 ~]$ $OGG_HOME/bin/ServiceManager
 --prop=/config/network/serviceListeningPort
 --value='{"port":9100,"address":"127.0.0.1"}' --type=array --persist --exit
```

Step 3.7.3 - Restart the Service Manager and Deployment

As the `grid` OS user on the first GGHub node, restart the GoldenGate deployment:

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ agctl start goldengate $GG_DEPLOYMENT_NAME
[grid@gghub_prim1 ~]$ agctl status goldengate

Goldengate  instance 'gghub1' is running on gghub_prim1
```

Step 3.7.4 - Modify the GoldenGate Microservices listener address

As the `oracle` OS user on the first GGHub node, modify all the GoldenGate microservices (adminsrvr, pmsrvr, distsrvr, recvsrvr) listening address to `localhost` for the deployments managed by the Service Manager using the following command:

```
[opc@gghub_prim1 ~]$ sudo chmod g+x /u01/oracle/scripts/secureServices.py
[opc@gghub_prim1 ~]$ sudo su - oracle
[oracle@gghub_prim1 ~]$ /u01/oracle/scripts/secureServices.py http://
localhost:9100
 --user oggadmin

Password for 'oggadmin': <oggadmin_password>

*** Securing deployment - gghub1

Current value of "/network/serviceListeningPort" for "gghub1/adminsrvr" is
9101
Setting new value and restarting service.
New value of "/network/serviceListeningPort" for "gghub1/adminsrvr" is
{
    "address": "127.0.0.1",
    "port": 9101

}.
Current value of "/network/serviceListeningPort" for "gghub1/distsrvr" is 9102
Setting new value and restarting service.
New value of "/network/serviceListeningPort" for "gghub1/distsrvr" is
{
    "address": "127.0.0.1",
    "port": 9102
}.
Current value of "/network/serviceListeningPort" for "gghub1/pmsrvr" is 9104
Setting new value and restarting service.
New value of "/network/serviceListeningPort" for "gghub1/pmsrvr" is
{
    "address": "127.0.0.1",
    "port": 9104
}.
Current value of "/network/serviceListeningPort" for "gghub1/recvsrvr" is 9103
Setting new value and restarting service.
New value of "/network/serviceListeningPort" for "gghub1/recvsrvr" is
{
    "address": "127.0.0.1",
    "port": 9103
}.
```

> **Note:**
>
> To modify a single deployment (adminsrvr, pmsrvr, distsrvr, recvsrvr), add the flag `--deployment instance_name`

Step 3.8 - Create a Clusterware Resource to Manage NGINX

Oracle Clusterware needs to have control over starting the NGINX reverse proxy so that it can be started automatically before the GoldenGate deployments are started.

As the `root` OS user on the first GGHub node, use the following command to create a Clusterware resource to manage NGINX. Replace `HOSTING_MEMBERS` and `CARDINALITY` to match your environment:

```
[root@gghub_prim1 ~]# sh /u01/oracle/scripts/add_nginx.sh

######################
List of VIP resources:
----------------------
gghub_prim1_vip1
----------------------
Application VIP CRS Resource: <gghub_prim1_vip1>
----------------------

########################
List of Hosting Members
------------------------
gghub_prim1
gghub_prim2
------------------------
HOSTING_MEMBERS: gghub_prim1,gghub_prim2
```

The NGINX resource created in this example will run on the named database nodes simultaneously, specified by `HOSTING_MEMBERS`. This is recommended when multiple GoldenGate Service Manager deployments are configured and can independently move between database nodes.

Once the NGINX Clusterware resource is created, the GoldenGate XAG resources need to be altered so that NGINX must be started before the GoldenGate deployments are started.

As the `root` OS user on the first GGHub node, modify the XAG resources using the following example commands.

```
# Determine the current --file systems parameter:

[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ agctl config goldengate $GG_DEPLOYMENT_NAME
 |grep -i "file system"

File System resources needed: acfs_primary

# Modify the --file systems parameter:

[opc@gghub_prim1 ~]$ /u01/app/grid/xag/bin/agctl modify goldengate
```

```
 $GG_DEPLOYMENT_NAME
 --filesystems acfs_primary,nginx

# Validate the current --file systems parameter:

[grid@gghub_prim1 ~]$ agctl config goldengate $GG_DEPLOYMENT_NAME
 |grep -i "File system"

File System resources needed: acfs_primary,nginx
```

> **Note:**
>
> - Repeat the above commands for each XAG GoldenGate registration relying on NGINX.
> - Repeat all the steps in step 3.8 for the primary and standby GGHub systems.

Step 3.9 - Create an Oracle Net TNS Alias for Oracle GoldenGate Database Connections

To provide local database connections for the Oracle GoldenGate processes when switching between nodes, create a TNS alias on **all** nodes of the cluster where Oracle GoldenGate may be started. Create the TNS alias in the `tnsnames.ora` file in the `TNS_ADMIN` directory specified in the deployment creation.

If the source database is a multitenant database, two TNS alias entries are required, one for the container database (CDB) and one for the pluggable database (PDB) that is being replicated. For a target Multitenant database, the TNS alias connects the PDB to where replicated data is being applied. The pluggable database `SERVICE_NAME` should be set to the database service created in an earlier step (refer to Step 2.3: Create the Database Services in Task 2: Prepare a Primary and Standby Base System for GGHub).

As the `oracle` OS user on any database node of the primary and the standby database systems, use `dbaascli` to find the database domain name and the SCAN name:

```
# Primary DB
[opc@exadb1_node1]$ sudo su - oracle
[oracle@exadb1_node1]$ source <dbName>.env
[oracle@exadb1_node1]$ dbaascli database getDetails --dbname <dbName> |grep
'connectString'

        "connectString" : "<primary_scan_name>:1521/<service_name>"

# Standby DB

[opc@exadb2_node1]$ sudo su - oracle
[oracle@exadb2_node1]$ source dbName.env
[oracle@exadb2_node1]$ dbaascli database getDetails --dbname <dbName> |grep
'connectString'

      "connectString" : "<standby_scan_name>:1521/<service_name>"
```

As the `oracle` OS user on all nodes of the primary and standby GGHub, add the recommended parameters for Oracle GoldenGate in the `sqlnet.ora` file:

```
[opc@gghub_prim1]$ sudo su - oracle
[oracle@gghub_prim1]$ mkdir -p /u01/app/oracle/goldengate/network/admin
[oracle@gghub_prim1]$
cat > /u01/app/oracle/goldengate/network/admin/sqlnet.ora <<EOF

DEFAULT_SDU_SIZE = 2097152
EOF
```

As the `oracle` OS user on all nodes of the primary and standby GGHub, follow the steps to create the TNS alias definitions:

```
[opc@gghub_prim1 ~]$ sudo su - oracle
[oracle@gghub_prim1 ~]$

cat > /u01/app/oracle/goldengate/network/admin/tnsnames.ora <<EOF

# Source
<source_cbd_service_name>=
     (DESCRIPTION =
         (CONNECT_TIMEOUT=3)(RETRY_COUNT=2)(LOAD_BALANCE=off)(FAILOVER=on)
(RECV_TIMEOUT=30)
         (ADDRESS_LIST =
             (ADDRESS = (PROTOCOL = TCP)(HOST=<primary_scan_name>)
(PORT=1521)))
         (ADDRESS_LIST =
             (ADDRESS = (PROTOCOL = TCP)(HOST=<standby_scan_name>)
(PORT=1521)))
         (CONNECT_DATA=(SERVICE_NAME =
<source_cbd_service_name>.goldengate.com)))

<source_pdb_service_name>=
    (DESCRIPTION =
        (CONNECT_TIMEOUT=3)(RETRY_COUNT=2)(LOAD_BALANCE=off)(FAILOVER=on)
(RECV_TIMEOUT=30)
        (ADDRESS_LIST =
            (ADDRESS = (PROTOCOL = TCP)(HOST=<primary_scan_name>)(PORT=1521)))
        (ADDRESS_LIST =
            (ADDRESS = (PROTOCOL = TCP)(HOST=<standby_scan_name>)(PORT=1521)))
        (CONNECT_DATA=(SERVICE_NAME =
<source_pdb_service_name>.goldengate.com)))

# Target
<target_pdb_service_name>=
    (DESCRIPTION =
        (CONNECT_TIMEOUT=3)(RETRY_COUNT=2)(LOAD_BALANCE=off)(FAILOVER=on)
(RECV_TIMEOUT=30)
        (ADDRESS_LIST =
            (ADDRESS = (PROTOCOL = TCP)(HOST=<primary_scan_name>)(PORT=1521)))
        (ADDRESS_LIST =
            (ADDRESS = (PROTOCOL = TCP)(HOST=<standby_scan_name>)(PORT=1521)))
        (CONNECT_DATA=(SERVICE_NAME =
<target_pdb_service_name>.goldengate.com)))
```

```
EOF

[oracle@gghub_prim1 ~]$ scp /u01/app/oracle/goldengate/network/admin/*.ora
 gghub_prim2:/u01/app/oracle/goldengate/network/admin
```

> **Note:**
>
> When the `tnsnames.ora` or `sqlnet.ora` (located in the `TNS_ADMIN` directory for the Oracle GoldenGate deployment) are modified, the deployment needs to be restarted to pick up the changes.

Task 4: Configure the Oracle GoldenGate Environment

Perform the following steps to complete this task:

- Step 4.1 - Create the Database Credentials
- Step 4.2 - Set Up Schema Supplemental Logging
- Step 4.3 - Create the Autostart Profile
- Step 4.3 - Configure Oracle GoldenGate Processes

Step 4.1 - Create the Database Credentials

With the Oracle GoldenGate deployment created, use the Oracle GoldenGate Administration Service home page to create the database credentials using the above TNS alias names.

As the `oggadmin` user, create the database credentials:

1. Log in into the Administration Service: https://gghub.example.com:443/*deployment_name*/adminsrvr
2. Click **Configuration** under **Administration Service**.
3. Click the **plus (+)** to **Add Credentials** under the **Database** tab.
4. Add the required information for the source and target CDB and PDB:

Data Center	Container	Domain	Alias	User ID
DC 1	CDB	GoldenGate	DC1_CDB	c##ggadmin@<tns_alias>
DC 1	PDB	GoldenGate	DC1_PDB	ggadmin@<tns_alias>
DC 2	CDB	GoldenGate	DC2_CDB	c##ggadmin@<tns_alias>
DC 2	PDB	GoldenGate	DC2_PDB	ggadmin@<tns_alias>

Step 4.2 - Setup Schema Supplemental Logging

- Log in to the Oracle GoldenGate Administration Server.
- Click **Configuration** under **Administration Service**.

- Click the **Connect to database** button under **Actions** for the **Source Database (Reg_CDB)**.
- Click the plus button (Add TRANDATA) to **Add TRANDATA** for the **Schema** or **Tables**.

Step 4.3 - Create the Autostart Profile

Create a new profile to automatically start the Extract and Replicat processes when the Oracle GoldenGate Administration Server is started. Then, restart if any Extract or Replicat processes are abandoned. With GoldenGate Microservices, auto start and restart is managed by Profiles.

Using the Oracle GoldenGate Administration Server GUI, create a new profile that can be assigned to each of the Oracle GoldenGate processes:

1. Log in to the **Administration Service** on the Source and Target GoldenGate.
2. Click on **Profile** under **Administration Service**.
3. Click the **plus (+)** sign next to Profiles on the Managed Process Settings home page.
4. Enter the details as follows:
    - Profile Name: Start_Default
    - Description: Default auto-start/restart profile
    - Default Profile: Yes
    - Auto Start: Yes
    - Auto Start Options
        - Startup Delay: 1 min
        - Auto Restart: Yes
    - Auto Restart Options
        - Max Retries: 5
        - Retry Delay: 30 sec
        - Retries Window: 30 min
        - Restart on Failure only: Yes
        - Disable Task After Retries Exhausted: Yes
5. Click **Submit**

Step 4.4 - Configure Oracle GoldenGate Processes

When creating Extract, Distribution Paths, and Replicat processes with Oracle GoldenGate Microservices Architecture, all files that need to be shared between the GGHub nodes are already shared with the deployment files stored on a shared file system.

Below are essential configuration details recommended for running Oracle GoldenGate Microservices on GGHub for Extract, Distribution Paths, and Replicat processes.

Perform the following sub-steps to complete this step:

- Step 4.4.1 - Extract Configuration
- Step 4.4.2 - Replicat Configuration
- Step 4.4.3 - Distribution Path Configuration
- Step 4.4.4 - Set Up a Heartbeat Table for Monitoring Lag Times

The main goal is to prevent data divergence between GoldenGate replicas and their associated standby databases. This section focuses on configuring Extract so that GoldenGate Extract never gets ahead of the standby database which can result in data divergence.

GoldenGate Parameter	Description	Recommendations
`TRANLOGOPTIONS HANDLEDLFAILOVER`	This is mandatory setting for Data Guard configurations that have Oracle GoldenGate to ensure GoldenGate Extract never extract data that has not been received by standby database. The HANDLEDLFAILOVER stands for handle DATA LOSS for Data Guard failover. The following parameter must be added to the Extract process parameter fileto avoid losing transactions and resulting in logical data inconsistencies after data loss Data Guard failover event. When the two primary tried to reconcile, this parameter ensures that all transactions can be reconciled since the new primary (old standby) is not further behind as expected. Prevents Extract from extracting redo data from the source database, and writing to the trail file data that has not yet been applied to the Oracle Data Guard standby database. If this parameter is not specified, after a data loss failover, it is possible to have data in the target database that is not present in the source database, leading to data divergence and logical data inconsistencies.	MANDATORY when the source database is configured with Data Guard in Max Availaibility or Max Performance mode.
`TRANLOGOPTIONS FAILOVERTARGETDESTID n`	For multiple standby configurations or cases when Data Guard Fast-Start failover is not enabled, set FAILOVERTARGETDESTID to standby demarcated by LOG_ARCHIV_DEST to ensure GoldenGate Extract never extract data that has not been received by target standby database. To determine the correct value for FAILOVERTARGETDESTID, use the LOG_ARCHIVE_DEST_N parameter from the GoldenGate source database which is used for sending redo to the source standby database. For example, if LOG_ARCHIVE_DEST_2 points to the standby database, then use a value of 2. When not using Data Guard Fast Start Failover (FSFO) in the source database, this parameter Identifies which standby database the Extract process must remain behind, with regard to not extracting redo data that has not yet been applied to the Oracle Data Guard standby database.	MANDATORY when not using FSFO in the source database. To determine the correct value for FAILOVERTARGETDESTID, use the LOG_ARCHIVE_DEST_N parameter from the GoldenGate source database which is used for sending redo to the source standby database. For example, if LOG_ARCHIVE_DEST_2 points to the standby database, then use a value of 2.

GoldenGate Parameter	Description	Recommendations
`TRANLOGOPTIONS HANDLEDLFAILOVER STANDBY_WARNING value`	The amount of time before a warning message is written to the Extract report file, if Extract is stalled, due to being unable to query the source database standby apply progress. This can occur after a Data Guard failover when the old primary database is not currently available. The default is 60 seconds.	OPTIONAL if want to adjust the timing of when the warning message is written to the Extract report file. Add STANDBY_WARNING <value> to the TRANLOGOPTIONS HANDLEDLFAILOVER parameter.
`TRANLOGOPTIONS HANDLEDLFAILOVER STANDBY_ABEND value`	The amount of time before Extract abends, if Extract is stalled, due to being unable to query the standby apply progress. The default is 30 minutes.	OPTIONAL if want to adjust the amount of time it takes Extract to abend, when the source database standby is not accessible to enforce the HANDLEDLFAILOVER parameter. Add STANDBY_ABEND <value> to the TRANLOGOPTIONS HANDLEDLFAILOVER parameter.
`TRANLOGOPTIONS DLFAILOVER_TIMEOUT value`	The amount of time Extract will run on the new source primary database, after a Data Guard role transition, before it will check the status of the standby database. If standby database is not available after the DLFAILOVER_TIMEOUT, Extract will abend. The default is 300 seconds. NOTE: If during normal operations of the source Oracle Data Guard configuration, the standby database becomes unavailable, Extract will stop extracting data from the source database to prevent possible data divergence with the GoldenGate target database due to the HANDLEDLFAILOVER parameter. The DLFAILOVER_TIMEOUT parameter does not take effect when a Data Guard failover has not occurred, and there are no messages output to the Extract report file.	OPTIONAL if you want to adjust the amount of time an Extract can run on a new primary source database, after a role transition, when the standby is not yet available to honor the TRANLOGOPTIONS HANDLEDLFAILOVER parameter.

Refer to the Reference for Oracle GoldenGate for more information about the Extract `TRANLOGOPTIONS` parameters.

When creating an Extract using the Oracle GoldenGate Administration Service GUI, leave the `Trail SubDirectory` parameter blank so that the trail files are automatically created in the deployment directories stored on the shared file system. The default location for trail files is the /<deployment directory>/var/lib/data directory.

> **Note:**
>
> To capture from a multitenant database, you must use an Extract configured at the root level using a c## account. To apply data into a multitenant database, a separate Replicat is needed for each PDB because a Replicat connects at the PDB level and doesn't have access to objects outside of that PDB.

Step 4.4.1 - Extract Configuration

1. Log in to the Oracle GoldenGate **Administration Server**
2. Click in **Overview** under **Administration Service**
3. Click the **plus (+)** button to **Add Extract**
4. Select **Integrated Extract**
5. Add the required information as follows:
   - Process Name: EXT_1
   - Description: Extract for DC 1 CDB
   - Intent: Unidirectional
   - Begin: Now
   - Trail Name: aa
   - Credential Domain: GoldenGate
   - Credential Alias: DC1_CDB
   - Register to PDBs: PDB Name
6. Click **Next** and set parameters:

```
EXTRACT ext_1
USERIDALIAS DC1_CDB DOMAIN GoldenGate
EXTTRAIL aa
TRANLOGOPTIONS HANDLEDLFAILOVER
TRANLOGOPTIONS FAILOVERTARGETDESTID 2
SOURCECATALOG <PDB_NAME>
TABLE <OWNER>.*;
```

7. Click **Next**.
8. If using CDB Root Capture from PDB, add the `SOURCECATALOG` parameter with the PDB Name
9. For Oracle Data Guard configurations, add the `TRANLOGOPTIONS` parameter, if required, as explained earlier in this step:
   - Add the parameter `TRANLOGOPTIONS HANDLEDLFAILOVER`
   - Add the parameter `TRANLOGOPTIONS FAILOVERTARGETDESTID <log_archive_dest_numer>` only if Oracle Data Guard Fast-Start Failover (FSFO) is NOT in use.
10. Click **Create and Run**.

See Oracle GoldenGate Extract Failure or Error Conditions Considerations for more information.

Step 4.4.2 - Replicat Configuration

Oracle generally recommends using integrated parallel Replicat which offers better apply performance for most workloads when the GGHub is in the same region as the target Oracle GoldenGate database.

The best apply performance can be achieved when the network latency between the GGHub and the target database is as low as possible. The following configuration is recommended for the remote Replicat running on the Oracle GGHub.

- `APPLY_PARALLELISM` – Disables automatic parallelism, instead of using `MAX_APPLY_PARALLELISM` and `MIN_APPLY_PARALLELISM`, and allows the highest amount of concurrency to the target database. It is recommended to set this as high as possible based on available CPU of the hub and the target database server.
- `MAP_PARALLELISM` – Should be set with a value of 2 to 5. With a larger number of appliers, increasing the Mappers increases the ability to hand work to the appliers.
- `BATCHSQL` – applies DML using array processing which reduces the amount network overheads with a higher latency network. Be aware that if there are many data conflicts, `BATCHSQL` results in reduced performance, as rollback of the batch operations followed by a re-read from trail file to apply in non-batch mode.

After you've set up your database connections and verified them, you can add a Replicat for the deployment by following these steps:

1. Log in to the Oracle GoldenGate **Administration Server**
2. Click theplus (+) sign next to **Replicats** on the Administration Service home page. The Add Replicat page is displayed.
3. Select a Replicat type and click **Next**.
4. Enter the details as follows:
   - Process Name: REP_1
   - Description: Replicat for DC 2 PDB
   - Intent: Unidirectional
   - Credential Domain: GoldenGate
   - Credential Alias: DC2_PDB
   - Source: Trail
   - Trail Name: aa
   - Begin: Position in Log
   - Checkpoint Table: "GGADMIN"."CHKP_TABLE"
5. Click **Next**
6. From the **Action Menu**, click **Details** to edit the Replicat **Parameters**:

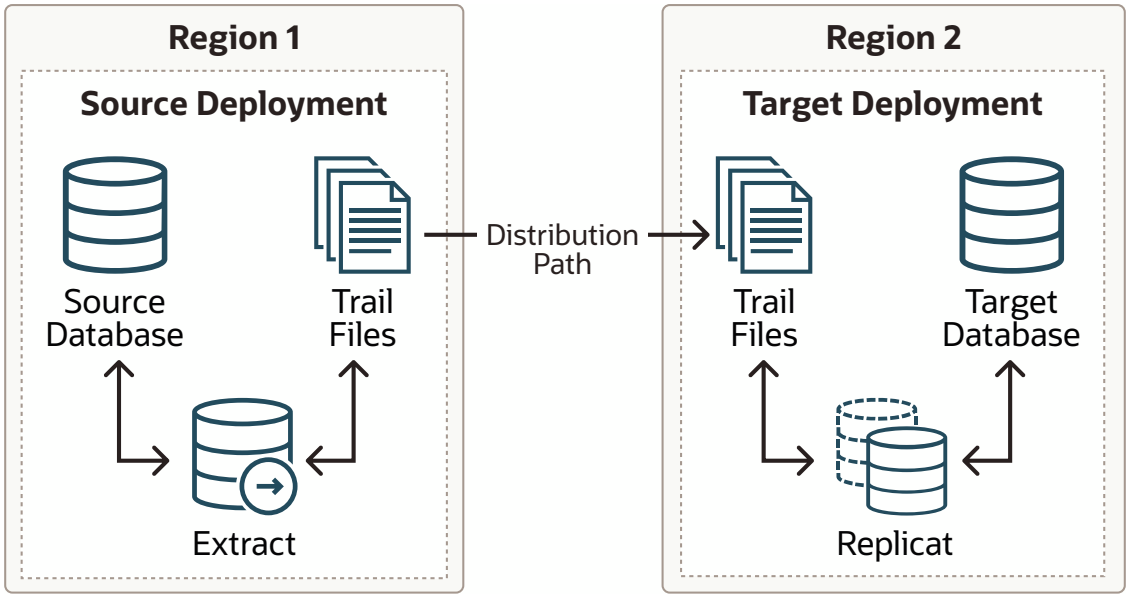
```
REPLICAT REP_1
USERIDALIAS Reg2_PDB DOMAIN GoldenGate
MAP <SOURCE_PDB_NAME>.<OWNER>.*, TARGET <OWNER>.*;
```

7. From the **Action Menu**, click **Start**.

Step 4.4.3 - Distribution Path Configuration

Distribution paths are only necessary when trail files need to be sent to an additional Oracle GoldenGate Hub in a different, or even the same, data center as described in the following figure.

Figure 23-4 Oracle GoldenGate Distribution Path



When using Oracle GoldenGate Distribution paths with the NGINX Reverse Proxy, additional steps must be carried out to ensure the path client and server certificates are configured.

More instructions about creating distribution paths are available in *Using Oracle GoldenGate Microservices Architecture*. A step-by-step example is in the following video, "Connect an on-premises Oracle GoldenGate to OCI GoldenGate using NGINX," to correctly configure the certificates.

Here are the steps performed in this sub-step:

- Step 4.4.3.1 - Download the Target Server's Root Certificate, and then upload it to the source Oracle GoldenGate
- Step 4.4.3.2 - Create a user in the Target Deployment for the Source Oracle GoldenGate to use
- Step 4.4.3.3 - Create a Credential in the Source Oracle GoldenGate
- Step 4.4.3.4 - Create a Distribution Path on the Source Oracle GoldenGate to the Target Deployment
- Step 4.4.3.5 - Distribution Path Recommendations

Step 4.4.3.1 - Download the Target Server's Root Certificate, and then upload it to the source Oracle GoldenGate

Download the target deployment server's root certificate and add the CA certificate to the source deployment Service Manager.

1. Log in to the **Administration Service** on the Target GoldenGate deployment.
2. Follow "Step 2 - Download the target server's root certificate" in the video "Connect an on-premises Oracle GoldenGate to OCI GoldenGate using NGINX."

Step 4.4.3.2 - Create a user in the Target Deployment for the Source Oracle GoldenGate to use

Create a user in the target deployment for the distribution path to connect to:

1. Log in to the **Administration Service** on the Target GoldenGate.
2. Click on **Administrator** under **Administration Service**.
3. Click the plus (+) sign next to **Users**.
4. Enter the details as follows:
   - Username: ggnet
   - Role: Operator
   - Type: Password
5. Click **Submit**

Step 4.4.3.3 - Create a Credential in the Source Oracle GoldenGate Deployment

Create a credential in the source deployment connecting the target deployment with the user created in the previous step. For example, a domain of OP2C and an alias of WSSNET.

1. Log in to the **Administration Service** on the Source Oracle GoldenGate.
2. Click in **Configuration** under **Administration Service.**
3. Click the **plus (+)** sign next to **Credentials** on the Database home page.
4. Enter the details as follows:
   - Credential Domain: OP2C
   - Credential Alias: wssnet
   - User ID: ggnet
5. Click **Submit**

Step 4.4.3.4 - Create a Distribution Path on the Source Oracle GoldenGate to the Target Deployment

A path is created to send trail files from the Distribution Server to the Receiver Server. You can create a path from the Distribution Service. To add a path for the source deployment:

1. Log in to the **Distribution Service** on the Source Oracle Goldengate.
2. Click the plus (+) sign next to **Path** on the Distribution Service home page. The Add Path page is displayed.
3. Enter the details as follows:

Option	Description
Path Name	Select a name for the path.
Source: *Trail Name*	Select the Extract name from the drop-down list, which populates the trail name automatically. If it doesn't, enter the trail name you provided while adding the Extract.
Generated Source URI	Specify `localhost` for the server's name; this allows the distribution path to be started on any of the Oracle RAC nodes.
Target Authentication Method	Use 'UserID Alias'
Target	Set the **Target** transfer protocol to wss (secure web socket). Set the **Target Host** to the target hostname/VIP that will be used for connecting to the target system along with the **Port Number** that NGINX was configured with (default is 443).

Option	Description
Domain	Set the **Domain** to the credential domain created above, for example, OP2C.
Alias	The **Alias** is set to the credential alias wssnet.
Auto Restart Options	Set the distribution path to restart when the Distribution Server starts automatically. This is required, so that manual intervention is not required after a RAC node relocation of the Distribution Server. It is recommended to set the number of **Retries** to 10. Set the **Delay**, which is the time in minutes to pause between restart attempts, to 1.

4. Click **Create Path**.
5. From the Action Menu, click **Start**.

Step 4.4.3.5 - Distribution Path Recommendations

If there are any GoldenGate distribution paths sending trail files to the GGHub, after a role transition of the GGHub, the paths will need to be altered to send the trail files to the new primary GGHub system. This can be done using the following example REST call:

```
curl -s -K src_access.cfg
 https://Source_VIP/Source_Deployment_Name/distsrvr/services/v2/sources/
Distribution_Path_Name
 -X PATCH --data '{"target":{"uri":"ogg://Target_VIP:9103/services/v2/targets?
trail=dd"}}' | python
 -m json.tool
```

You can automate changing the source distribution path target address after a hub role transition using the sample shell script shown in Managing Planned and Unplanned Outages for Oracle GoldenGate Hub which is called by the `acfs_standby` CRS action script when a file system switchover/failover occurs.

The source distribution paths must be configured to restart automatically after it has failed so that if the target GoldenGate deployment relocates between Oracle RAC nodes or to the standby hub, the distribution paths will restart. If a distribution path was created without automatic restart enabled, it can be enabled through the distribution server web UI or a REST call. For example:

```
$ curl -s -K
 access.cfg https://<Source VIP>/<Source Deployment Name>/distsrvr/
services/v2/sources/ggs_to_gghub
 -X PATCH --data '{"options":{"autoRestart":{"delay": 2,"retries": 10}}}' |
python -m json.tool
```

To check the current configuration of a distribution path, use the following example:

```
$ curl -s -K
 access.cfg https://<Source VIP>/<Source Deployment Name>/distsrvr/
services/v2/sources/ggs_to_gghub
 -X GET | python -m json.tool

# Sample output:
"name": "scam_to_gghub",
  "options": {
     "autoRestart": {
```

```
    "delay": 2,
    "retries": 10
},
```

Step 4.4.4 - Set up a Heartbeat Table for Monitoring Lag Times

Use the instructions in Steps to add Heartbeat Table in OCI GoldenGate to implement the best practices for creating a heartbeat process that can be used to determine where and when lag is developing between a source and target system.

This document guides you through the step-by-step process of creating the necessary tables and added table mapping statements needed to keep track of processing times between a source and target database. Once the information is added into the data flow, the information is then stored into a target tables that can be analyzed to determine when and when the lag is introduced between the source and target systems.

24

On-Premises: Oracle GoldenGate Microservices Architecture with Oracle Real Application Clusters Configuration Best Practices

Use these best practices for configuring Oracle GoldenGate Microservices Architecture for on-premises systems, including Oracle Exadata, to work with Oracle Real Application Clusters (RAC), Oracle Clusterware, and Oracle Database File System (DBFS) or Oracle Advanced Cluster File System (ACFS).

The target Oracle RAC system that hosts Oracle GoldenGate Microservices Architecture can act as the source database, as the target database, or in some cases as both source and target databases, for Oracle GoldenGate operations.

See the following topics:

Summary of Recommendations when Deploying Oracle GoldenGate on Oracle RAC

When configuring Oracle GoldenGate in an Oracle RAC environment, follow these recommendations.

- Install the latest version of Oracle GoldenGate software locally on each Oracle RAC node, making sure that the software location is the same on all Oracle RAC nodes.
- Use the Oracle Database File System (DBFS) or Oracle Advanced Cluster File System (ACFS) for the file system where the Oracle GoldenGate files are stored (trail, checkpoint, temporary, report, and parameter files).

- Use the same DBFS or ACFS mount point on all of the Oracle RAC nodes that may run Oracle GoldenGate.
- When creating the GoldenGate deployment, specify either DBFS or ACFS for the deployment location.
- Install Grid Infrastructure agent (XAG) version 10 or later on all Oracle RAC nodes that will run Oracle GoldenGate.
- Configure the GoldenGate processes to automatically start and restart when the deployment is started.

Task 1: Configure the Oracle Database for Oracle GoldenGate

The source and target Oracle GoldenGate databases should be configured using the following recommendations.

- Enable Oracle GoldenGate replication by setting the database initialization parameter `ENABLE_GOLDENGATE_REPLICATION=TRUE`.
- Run the Oracle GoldenGate source database in `ARCHIVELOG` mode.
- Enable `FORCE LOGGING` mode in the Oracle GoldenGate source database.
- Enable minimal supplemental logging in the source database. Additionally, add schema or table level logging for all replicated objects.
- If the Replicat process will be used, configure the streams pool in the System Global Area (SGA) on the source database using the `STREAMS_POOL_SIZE` initialization parameter.

  Note that the streams pool is only needed on the target database if integrated Replicat will be used.

  Use the following equation to determine the value for `STREAMS_POOL_SIZE`:

  `STREAMS_POOL_SIZE` = (#Extracts and #Integrated Replicats * 1GB) * 1.25

  For example, in a database with 2 Extracts and 2 integrated Replicats:

  `STREAMS_POOL_SIZE` = 4GB * 1.25 = 5GB

  When adding Extract or Replicat processes, it is important to recalculate and configure the new streams pool size requirement.

For more information about preparing the database for Oracle GoldenGate, see Preparing the Database for Oracle GoldenGate.

Task 2: Create the Database Replication Administrator User

The source and target Oracle databases need a GoldenGate Administrator user with appropriate privileges assigned.

For single tenant (non-CDB architecture) databases, see Establishing Oracle GoldenGate Credentials

For a multitenant source database, GoldenGate Extract must be configured to connect to a user in the root container database, using a `c##` account. For a multitenant target database, a separate GoldenGate administrator user is needed for each PDB that a Replicat applies data to.

For more details about creating a GoldenGate Administrator in an Oracle Multitenant Database, see Configuring Oracle GoldenGate in a Multitenant Container Database

Task 3: Create the Database Services

A database service is required so that the Oracle Grid Infrastructure Agent automatically starts the GoldenGate deployment when the database is opened. When DBFS is used for the shared file system, the database service is also used to mount DBFS to the correct Oracle RAC instance.

When using a source multitenant database, a separate service is required for the root container database (CDB) and the pluggable database (PDB) that contains the schema being replicated. For a target multitenant database, a single service is required for the PDB.

Create the service using the following command, as the `oracle` user.

```
$ srvctl add service -db db_name -service service_name
 -preferred instance_1 -available instance_2, instance_3 etc.
 -pdb PDB_name
```

For example:

```
$ srvctl add service -db ggdb -service oggserv_pdb -preferred ggdb1
    -available ggdb2 -pdb GGPDB01
```

It you are not using Oracle Multitenant Database, omit the `-pdb` parameter.

Task 4: Set Up a File System on Oracle RAC

Oracle GoldenGate Microservices Architecture is designed with a simplified installation and deployment directory structure. The installation directory should be placed on local storage on each Oracle RAC node to provide minimized downtime during software patching.

The deployment directory, which is created during deployment creation using the Oracle GoldenGate Configuration Assistant (oggca.sh), must be placed on a shared file system. The deployment directory contains configuration, security, log, parameter, trail, and checkpoint files.

Placing the deployment in DBFS or ACFS provides the best recovery and failover capabilities in the event of a system failure. Ensuring the availability of the checkpoint files cluster-wide is essential so that after a failure occurs the GoldenGate processes can continue running from their last known position.

If Oracle GoldenGate will be configured along with Oracle Data Guard, the recommended file system is DBFS. DBFS is contained in the database protected by Data Guard, and can be fully integrated with XAG. In the event of a Data Guard role transition, the file system can be automatically mounted on the new primary server, followed by automated start-up of Oracle GoldenGate. This is currently not possible with ACFS, because it is not part of the Oracle Data Guard configuration.

Follow the instructions in the appropriate section below to configure the file system for either DBFS or ACFS.

Oracle Database File System (DBFS)

It is required that you create the DBFS tablespace inside the same database that the Oracle GoldenGate processes are connected to. For example, if a GoldenGate integrated Extract

process is extracting from a database called GGDB, the DBFS tablespace would be located in the same GGDB database.

Follow instructions in My Oracle Support note 869822.1 to install the required FUSE libraries if they are not already installed.

Use the instructions in My Oracle Support note 1054431.1 to configure the database, tablespace, database user, `tnsnames.ora` Oracle Net connection alias, and permissions on source or target GoldenGate environments required for DBFS.

> **Note:**
>
> When using an Oracle Multitenant Database, the DBFS tablespace MUST be created in a Pluggable Database (PDB). It is recommended that you use the same PDB that the GoldenGate Extract or Replicat processes are connecting to, allowing DBFS to use the same database service, created above in Task 2, for its database dependency.

When you create a file system for storing the GoldenGate deployment files, it is recommended that you allocate enough trail file disk space to permit storage of up to 12 hours of trail files. Doing this provides sufficient space for trail file generation should a problem occur with the target environment that prevents it from receiving new trail files. The amount of space needed for 12 hours can only be determined by testing trail file generation rates with real production data.

Example DBFS creation:

```
$ cd $ORACLE_HOME/rdbms/admin
$ sqlplus dbfs_user/dbfs_password@database_tns_alias
SQL> start dbfs_create_filesystem dbfs_gg_tbs goldengate
```

Follow the instructions in My Oracle Support note 1054431.1 to configure the newly created DBFS file system so that the DBFS instance and mount point resources are automatically started by Cluster Ready Services (CRS) after a node failure, with the following DBFS configuration and script file modifications.

1. Change the `mount-dbfs.conf` parameters to reflect your database environment.

   Modify the `MOUNT_OPTIONS` parameter to the following:

   ```
   MOUNT_OPTIONS=allow_other,direct_io,failover,nolock
   ```

   The `failover` option forces all file writes to be committed to the DBFS database in an `IMMEDIATE WAIT` mode. This prevents data getting lost when it has been written into the `dbfs_client` cache but not yet written to the database at the time of a database or node failure.

   The `nolock` mount option is required if you are using Oracle Database 18c or a later release, due to a change in the DBFS file locking which can cause issues for GoldenGate processes after an Oracle RAC node failure when a file is currently locked.

   If you are using a `dbfs_client` from Oracle Database 12c Release 2 (12.2), make sure you have applied the latest release update that includes the fix for bug 27056711. Once the fix has been applied, the `MOUNT_OPTIONS` should also include the `nolock` option.

2. Modify the `mount-dbfs.sh` script to force unmounting of DBFS when the CRS resource is stopped.

   Change two occurrences of:

   ```
   $FUSERMOUNT -u $MOUNT_POINT
   ```

   To the following:

   ```
   $FUSERMOUNT -uz $MOUNT_POINT
   ```

3. When registering the resource with Oracle Clusterware, be sure to create it as a `cluster_resource` instead of a `local_resource`, as specified in the My Oracle Support note.

   The reason for using `cluster_resource` is so that the file system can only be mounted on a single node at a time, preventing mounting of DBFS from concurrent nodes, which creates the potential for concurrent file writes, causing file corruption problems.

   Make sure to use the database service name created in a previous step for the DBFS service dependency.

   For example:

   ```
   DBNAME=ggdb
   DEPNAME=ora.$DBNAME.oggserv.svc

   crsctl add resource $RESNAME \
     -type cluster_resource \
     -attr "ACTION_SCRIPT=$ACTION_SCRIPT, \
            CHECK_INTERVAL=30,RESTART_ATTEMPTS=10, \
   START_DEPENDENCIES='hard($DEPNAME)pullup($DEPNAME)',\
            STOP_DEPENDENCIES='hard($DEPNAME)',\
            SCRIPT_TIMEOUT=300"
   ```

   Once the DBFS resource has been created, the file system should be mounted and tested.

   ```
   $ crsctl start res dbfs_mount
   $ crsctl stat res dbfs_mount
   ```

   After the file system is mounted, create the directory for storing the GoldenGate files.

   ```
   $ cd /mnt/dbfs/goldengate
   $ mkdir deployments
   ```

> **Note:**
>
> Leave the shared file system mounted. It is required for creating the GoldenGate deployment in a later step.

Oracle Advanced Cluster File System (ACFS)

Oracle ACFS is an alternative to DBFS for the shared GoldenGate files in an Oracle RAC configuration.

Refer to My Oracle Support note 1929629.1 for more information about ACFS configuration requirements for Oracle Exadata Database Machine.

Create a single ACFS file system for storing the Oracle deployment files.

It is recommended that you allocate enough trail file disk space to permit storage of up to 12 hours of trail files. Doing this provides sufficient space for trail file generation should a problem occur with the target environment that prevents it from receiving new trail files. The amount of space needed for 12 hours can only be determined by testing trail file generation rates with real production data.

1. Create the file system using ASMCMD as the Oracle ASM administrator user.

```
ASMCMD [+] > volcreate -G datac1 -s 1200G ACFS_GG
```

> **Note:**
>
> Modify the file system size according to the determined size requirements.

```
ASMCMD> volinfo -G datac1 acfs_gg

Diskgroup Name: DATAC1
         Volume Name: ACFS_GG
         Volume Device: /dev/asm/acfs_gg-151
         State: ENABLED
         Size (MB): 1228800
         Resize Unit (MB): 64
         Redundancy: MIRROR
         Stripe Columns: 8
         Stripe Width (K): 1024
         Usage:
         Mountpath:
```

Make the file system with the following mkfs command.

```
$ /sbin/mkfs -t acfs /dev/asm/acfs-gg-151
```

2. Create the CRS resource for the newly created ACFS file system, if not already created.

Check to see if the file system resource was already created.

```
$ srvctl status filesystem -volume ACFS_GG -diskgroup DATAC1

ACFS file system /mnt/acfs_gg is mounted on nodes oggadm07, oggadm08
```

If not already created, create the ACFS mount point on all of the Oracle RAC nodes.

```
# mkdir -p /mnt/acfs_gg
```

Create the file system resource as the root user. Due to the implementation of distributed file locking on ACFS, unlike DBFS, it is acceptable to mount ACFS on more than one RAC node at any one time.

Create the ACFS resource using srvctl from the Oracle Grid Infrastructure ORACLE_HOME.

```
# srvctl add filesystem -device /dev/asm/acfs_gg-151 -volume ACFS_GG
-diskgroup DATAC1 -path /mnt/acfs_gg -user oracle -autostart RESTORE
```

To verify the currently configured ACFS file systems, use the following command to view the file system details.

```
$ srvctl config filesystem

Volume device: /dev/asm/acfs_gg-151
Diskgroup name: datac1
Volume name: ACFS_GG
Canonical volume device: /dev/asm/acfs_gg-151
Accelerator volume devices:
Mountpoint path: /mnt/acfs_gg
Mount point owner: oracle

Check the status of the ACFS resource and mount it.

$ srvctl status filesystem -volume ACFS_GG -diskgroup DATAC1

ACFS file system /mnt/acfs is not mounted

$ srvctl start filesystem -volume ACFS_GG -diskgroup DATAC1 -node
dc1north01
```

The CRS resource that is created is named using the format ora.*diskgroup_name*.*volume_name*.acfs. Using the above file system example, the CRS resource is called `ora.datac1.acfs_gg.acfs`.

To see all ACFS file system CRS resources that currently exist, use the following command.

```
$ crsctl stat res -w "((TYPE = ora.acfs.type) OR (TYPE =
ora.acfs_cluster.type))"

NAME=ora.datac1.acfs_gg.acfs
TYPE=ora.acfs.type
TARGET=ONLINE              , OFFLINE
STATE=ONLINE on dc1north01, OFFLINE
```

3. Create a GoldenGate deployment directory on ACFS.

   After the file system is mounted, create the directory for storing the GoldenGate deployments.

```
$ cd /mnt/acfs_gg
$ mkdir deployments
```

> **Note:**
>
> Leave the shared file system mounted. It is required for creating the GoldenGate deployment in a later Task.

Task 5: Install Oracle GoldenGate

Download and install the Oracle GoldenGate 21c Microservices software, or later release.

Download the software at https://www.oracle.com/middleware/technologies/goldengate-downloads.html.

Install the Oracle GoldenGate software **locally** on all nodes in the Oracle RAC configuration that will be part of the GoldenGate configuration. Make sure the installation directory is **identical** on all nodes.

Follow the generic installation instructions detailed in Oracle GoldenGate Microservices Documentation.

Task 6: Create the Oracle GoldenGate Deployment

Once the Oracle GoldenGate software has been installed, the next step is to create a deployment using the Oracle GoldenGate Configuration Assistant (`oggca`).

There are two limitations that currently exist with Oracle GoldenGate and XAG:

1. A Service Manager that is registered with XAG can only manage a single deployment. If multiple deployments are required, each deployment must use their own Service Manager. Oracle GoldenGate release 21c simplifies this requirement because it uses a single deployment to support Extract and Replicat processes connecting to different versions of the Oracle Database.
2. Each Service Manager registered with XAG must belong to separate `OGG_HOME` software installation directories. Instead of installing Oracle GoldenGate multiple times, the recommended approach is to install Oracle GoldenGate one time, and then create a symbolic link for each Service Manager `OGG_HOME`.

For example:

```
$ echo $OGG_HOME
/u01/oracle/goldengate/gg21c_MS

$ ln -s /u01/oracle/goldengate/gg21c_MS /u01/oracle/goldengate/
gg21c_MS_ggnorth

$ export OGG_HOME=/u01/oracle/goldengate/gg21c_MS_ggnorth
$ $OGG_HOME/bin/oggca.sh
```

The symbolic link and `OGG_HOME` environment variable must be configured **before** running the Oracle GoldenGate Configuration Assistant on all Oracle RAC nodes.

Recommendations for creating the GoldenGate deployment in the Oracle GoldenGate Configuration Assistant are as follows.

1. In Service Manager Options, specify the following for the creation of a new Service Manager.
   a. In the Service Manager Details pane, select **Create New Service Manager**.
   b. Enter the **Service Manager Deployment Home** location on the shared DBFS or ACFS file system.
   c. Select to **Integrate with XAG**.
   d. In the Service Manager Connection Details pane, specify `localhost` in the **Listening hostname/address** field.

      Using `localhost` allows the deployment to be started on all of the Oracle RAC nodes without the need for a Virtual IP address (VIP).
   e. Enter the port number in **Listening port**.
2. In **Deployment Directories**, specify the Deployment home directory on the shared DBFS or ACFS file system.
3. In **Environment Variables**, specify a correct **TNS_ADMIN** directory.

   Starting with Oracle GoldenGate release 21.3, a database `ORACLE_HOME` is no longer required because the required database libraries are installed as part of the Oracle GoldenGate installation. It is recommended that you use `TNS_ADMIN` directory outside of any existing `ORACLE_HOME` directories.
4. In Security Options, do **not** select **SSL/TLS Security**.

   External access to the Oracle GoldenGate Microservices server is achieved by using NGINX Reverse Proxy SSL-termination. Secure access and communication to the GoldenGate deployments will be exclusively through the SSL port 443. Internal connectivity within the same local host between NGINX and GoldenGate does not require SSL.
5. In **Port Settings**, if the Management Pack for Oracle GoldenGate has been licensed, select **Enable Monitoring** to use the performance metric server using either Berkeley Database (BDB) or Lightening Memory Database (LMDB).

   For both BDB and LMDB Metrics Service DataStore types, set the **Metrics Service DataStore home** directory to a local directory that exists on all Oracle RAC nodes. For example: `/u01/oracle/goldengate/datastores/`*`deployment name`*
6. Continue through the Oracle GoldenGate Configuration Assistant until the deployment is created.
7. After the deployment has been created, if you are using DBFS for the shared file system and the database version is a release earlier than Oracle Database Release 21c (21.3), run the following commands to move the Oracle GoldenGate deployment temp directory from DBFS to local storage.

   On the first node:

   ```
   $ cd <DBFS GoldenGate deployment home directory/var
   $ mkdir -p local_storage_directory/deployment_name
   $ mv temp local_storage_directory/deployment_name
   $ ln -s local_storage_directory/deployment_name/temp temp
   ```

   On all other nodes:

   ```
   $ mkdir local_storage_directory/deployment_name/temp
   ```

First node example:

```
$ cd /mnt/dbfs/goldengate/deployments/ggnorth/var
$ mkdir -p /u01/oracle/goldengate/deployments/ggnorth
$ mv temp /u01/oracle/goldengate/deployments/ggnorth
$ ln -s /u01/oracle/goldengate/deployments/ggnorth/temp temp
```

On all other nodes:

```
$ mkdir /u01/oracle/goldengate/deployments/ggnorth/temp
```

Task 7: Oracle Clusterware Configuration

The following procedure shows you how to configure Oracle Clusterware to manage Oracle GoldenGate using the Oracle Grid Infrastructure Standalone Agent (XAG).

Using XAG automates the mounting of the shared file system (DBFS or ACFS) and the stopping and starting of the GoldenGate deployment when relocating between Oracle RAC nodes.

1. Install the Oracle Grid Infrastructure Standalone Agent.

   It is recommended that you install the XAG software as a standalone agent outside of the Grid Infrastructure ORACLE_HOME. This allows you to use the latest XAG release available, and the software can be updated without impact to the Grid Infrastructure.

   When using Oracle GoldenGate Microservices Architecture you **MUST** use XAG version 10.2 or later.

   The latest agent software is available for download from the following location:

   http://www.oracle.com/technetwork/database/database-technologies/clusterware/downloads/xag-agents-downloads-3636484.html

   Install the XAG standalone agent outside of the Oracle Grid Infrastructure home directory. XAG must be installed in the same directory on all RAC nodes in the cluster where Oracle GoldenGate is installed.

   For example, as the Oracle Grid Infrastructure user, the default of oracle:

   ```
   $ ./xagsetup.sh --install --directory /u01/oracle/xag --all_nodes
   ```

   Add the location of the newly installed XAG software to the PATH variable so that the location of `agctl` is known when the `oracle` user logs on to the machine.

   ```
   $ cat .bashrc
   export PATH=/u01/oracle/xag/bin:$PATH
   ```

   > **Note:**
   >
   > It is important to make sure that the XAG bin directory is specified BEFORE the Grid Infrastructure bin directory, to ensure the correct agctl binary is found. Set this location in the oracle user environment to take effect at time of logging on, such as in the .bashrc file when the Bash shell is in use.

2. Prepare for Application Virtual IP Address (VIP) Creation.

   A dedicated application VIP is required to allow access to the GoldenGate Microservices using the same host name, regardless of which Oracle RAC node is hosting the services. An application VIP also ensures that the GoldenGate Distribution Server can communicate with the Distribution Receiver running the current Oracle RAC node.

   The VIP is a cluster resource that Oracle Clusterware manages. The VIP is assigned to a cluster node and is automatically migrated to another node in the event of a node failure.

   There are two pieces of information needed before creating the application VIP:

   - The network number, which can be identified using the following command.

     ```
     $ crsctl status resource -p -attr ADDRESS_TYPE,NAME,USR_ORA_SUBNET -w
     "TYPE = ora.network.type" |sort | uniq

     ADDRESS_TYPE=IPV4
     NAME=ora.net1.network

     USR_ORA_SUBNET=10.133.16.0
     ```

     The net1 in `NAME=ora.net1.network` indicates that this is network 1, and it is of type IPV4.

   - The IP address for the new Application VIP, provided by your system administrator. This IP address must be in the same subnet of the cluster environment as determined above.

     The VIP will be created in the next step, when configuring the Oracle Grid Infrastructure Agent.

3. Configure Oracle Grid Infrastructure Agent (XAG).

   Oracle GoldenGate must be registered with XAG so that the deployment is started and stopped automatically when the database is started and the file system is mounted.

   To register Oracle GoldenGate Microservices Architecture with XAG use the following command format.

   ```
   agctl add goldengate instance_name
     --gg_home GoldenGate_Home
     --service_manager
     --config_home GoldenGate_SvcMgr_Config
     --var_home GoldenGate_SvcMgr_Var_Dir
     --port port_number
     --oracle_home $OGG_HOME/lib/instantclient
     --adminuser OGG_admin_user
     --user GG_instance_user
     --group GG_instance_group
     --network network_number
     --ip ip_address
     --vip_name vip_name
     --filesystems CRS_resource_name
     --db_services service_name
     --use_local_services
     --nodes node1, node2, ... ,nodeN
   ```

   Where:

`--gg_home` specifies the location of the Oracle GoldenGate software. Specify the `OGG_HOME` symbolic link for the `OGG_HOME` if registering multiple Service Managers (see Task 6: Create the Oracle GoldenGate Deployment).

`--service_manager` indicates this is a GoldenGate Microservices instance.

`--config_home` specifies the GoldenGate Service Manager deployment configuration home directory.

`--var_home` specifies the GoldenGate Service Manager deployment variable home directory.

`--port` specifies the deployment Service Manager port number.

`--oracle_home` specifies the location of the Oracle database libraries that are included as part of Oracle GoldenGate 21c and later releases. Example: `$OGG_HOME/lib/instantclient`

`--adminuser` specifies the Oracle GoldenGate Microservices administrator account name.

`--user` specifies the name of the operating system user that owns the GoldenGate deployment.

`--group` specifies the name of the operating system group that owns the GoldenGate deployment.

`--network` specifies the network subnet for the VIP, determined above.

`--ip` specifies the IP address for the VIP, which was determined above. If you have already created a VIP, then specify it using the `--vip_name` parameter in place of `--network` and `--ip`.

`--vip_name` specifies a CRS resource name for an application VIP that has previously been created. This parameter replaces `--network` and `--ip` (optional).

`--filesystems` specifies the DBFS or ACFS CRS file system resource that must be mounted before the deployment is started.

`--db_services` specifies the `ora.`*`database`*`.`*`service_name`*`.svc` service name that was created in the previous step. If using Oracle Multitenant Database, specify the PDB database service for Replicat, or the CDB database service for an Extract. If using both Replicat and Extract, specify both services names, separated by a comma.

`--use_local_services` specifies that the GoldenGate instance must be co-located on the same Oracle RAC node where the `db_services` service is running.

`--nodes` specifies which of the Oracle RAC nodes this GoldenGate instance can run on. If GoldenGate is configured to run on any of the Oracle RAC nodes in the cluster, this parameter should still be used to determine the preferred order of nodes to run Oracle GoldenGate.

Notes:

- The GoldenGate instance registration with XAG **MUST** be run as the `root` user.
- The `user` and `group` parameters are mandatory because the GoldenGate registration with XAG is run as the `root` user.

Below are some examples of registering Oracle GoldenGate with XAG.

Example 1: Oracle RAC cluster using DBFS, using an already created application VIP

```
# agctl add goldengate GGNORTH \
--gg_home /u01/oracle/goldengate/gg21c_MS \
```

```
--service_manager \
--config_home /mnt/dbfs/goldengate/deployments/ggsm01/etc/conf \
--var_home /mnt/dbfs/goldengate/deployments/ggsm01/var \
--port 9100 \
--oracle_home /u01/oracle/goldengate/gg21c_MS/lib/instantclient
--adminuser oggadmin
--user oracle \
--group oinstall \
--vip_name gg_vip_prmy \
--filesystems dbfs_mount \
--db_services ora.ds19c.oggserv.svc \
--use_local_services \
--nodes dc1north01,dc1north02
```

Where:

- GoldenGate instance is GGNORTH
- GoldenGate home directory is `/u01/oracle/goldengate/gg21c_MS`
- This is an Oracle GoldenGate Microservices Architecture instance (`--service_manager`)
- GoldenGate deployment configuration home directory is `/mnt/dbfs/goldengate/deployments/ggsm01/etc/conf`
- GoldenGate deployment variable home directory is `/mnt/dbfs/goldengate/deployments/ggsm01/var`
- Deployment Service Manager port number is 9100
- Oracle GoldenGate Microservices administrator account name is `oggadmin`
- The GoldenGate user is `oracle` in the group `oinstall`
- Application VIP name, managed by CRS, is called `gg_vip_prmy`
- The CRS resource name for the file system the deployment depends on is `dbfs_mount`
- The GoldenGate instance will be started on the same Oracle RAC node as the CRS service called `ora.ds19c.oraserv.svc` will be co-located on the same node as this GoldenGate instance.

Example 2: Oracle RAC cluster, using ACFS, with an application VIP running on a subset of the nodes in the cluster.

```
# agctl add goldengate GGNORTH \
--gg_home /u01/oracle/goldengate/gg21c_MS \
--service_manager \
--config_home /mnt/acfs/goldengate/deployments/ggsm01/etc/conf \
--var_home /mnt/acfs/goldengate/deployments/ggsm01/var \
--port 9100 \
--oracle_home /u01/oracle/goldengate/gg21c_MS/lib/instantclient
--adminuser admin \
--user oracle \
--group oinstall \
--network 1 --ip 10.13.11.203 \
--filesystems ora.datac1.acfs_gg.acfs \
--db_services ora.ds19c.oraserv.svc \
--use_local_services \
--nodes dc1north01,dc1north02
```

Where:

- GoldenGate instance is GGNORTH
- GoldenGate home directory is `/u01/oracle/goldengate/gg21c_MS`
- This is an Oracle GoldenGate Microservices Architecture instance (`--service_manager`)
- GoldenGate deployment configuration home directory is `/mnt/acfs/goldengate/deployments/ggsm02/etc/conf`
- GoldenGate deployment variable home directory is `/mnt/acfs/goldengate/deployments/ggsm02/var`
- Deployment Service Manager port number is 9100
- Oracle GoldenGate Microservices administrator account name is `admin`
- GoldenGate user is `oracle` in the group `oinstall`
- The network is the default `ora.net1.network` and the VIP is `10.13.11.203`
- The CRS resource name for the file system the deployment depends on is `ora.datac1.acfs_gg.acfs`
- This GoldenGate instance will be started on the same Oracle RAC node as the CRS service called `ora.ds19c.oraserv.svc` will be co-located on the same node as this GoldenGate instance
- Oracle GoldenGate will only run on Oracle RAC nodes dc1north01 and dc1north02, listed in priority order.

Example AGCTL Commands

Below are some example agctl commands that are used to manage the Oracle GoldenGate deployment with XAG.

To check the status of Oracle GoldenGate:

```
% agctl status goldengate
Goldengate  instance 'GGNORTH' is running on dc1north01
```

To start the GoldenGate deployment, and all Extract/Replicat processes that have been configured to autostart (instructions in a later step):

```
% agctl start goldengate GGNORTH --node dc1north02
```

To stop the GoldenGate deployment:

```
% agctl stop goldengate GGNORTH
```

To manually relocate the GoldenGate deployment to another node:

```
% agctl relocate goldengate GGNORTH --node dc1north02
```

To view the configuration parameters for the GoldenGate resource:

```
% agctl config goldengate GGNORTH

Instance name: GGNORTH
```

```
Application GoldenGate location is: /u01/oracle/goldengate/gg21c_MS
Goldengate MicroServices Architecture environment: yes
Goldengate Service Manager configuration directory: /mnt/dbfs/goldengate/
deployments/ggsm01/etc/conf

Goldengate Service Manager var directory: /mnt/dbfs/goldengate/deployments/
ggsm01/var

Service Manager Port: 9100
Goldengate Administration User: oggadmin
Autostart on DataGuard role transition to PRIMARY: no
Configured to run on Nodes: dc1north01 dc1north02
ORACLE_HOME location is: /u01/oracle/goldengate/gg21c_MS/lib/instantclient
Database Services needed: ora.cdb1.oggcdb.svc [use_local_services]
File System resources needed: ora.datac1.acfs_gg.acfs
Network: 1, IP: 10.13.11.203, User:oracle, Group:oinstall
```

To delete the GoldenGate XAG resource:

```
$ agctl stop goldengate GGNORTH
# agctl remove goldengate GGNORTH
```

For more information about the Oracle Grid Infrastructure Bundled Agent see Oracle Grid Infrastructure Standalone Agents for Oracle Clusterware 11g Rel. 2, 12c, 18c and 19c.

Task 8: Configure NGINX Reverse Proxy

Follow the instructions provided in My Oracle Support note 2826001.1 to install and configure NGINX Reverse Proxy with SSL connection, and to ensure all external communication is completely secure.

> **Note:**
>
> When using CA Signed Certificates with NGINX, make sure the NGINX `ssl_certificate` parameter points to a certificate file that contains the certificates in the correct order of CA signed certificate, intermediate certificate and root certificate.

Oracle Clusterware needs to have control over starting the NGINX reverse proxy so that it can be started automatically before the GoldenGate deployments are started.

The NGINX resource is created with a dependency on the underlying network CRS resource, the name of which can be determined using the following command:

```
$ $GRID_HOME/bin/crsctl stat res -w "TYPE == ora.network.type"|grep NAME

NAME=ora.net1.network
```

As the `root` user, use the following example command to create a Clusterware resource to manage NGINX.

```
# $GRID_HOME/bin/crsctl add resource nginx -type generic_application -attr
 "ACL='owner:root:rwx,pgrp:root:rwx,other::r--,group:oinstall:r-
x,user:oracle:rwx',
EXECUTABLE_NAMES=nginx,START_PROGRAM='/bin/systemctl
 start -f nginx',STOP_PROGRAM='/bin/systemctl
 stop -f nginx',CHECK_PROGRAMS='/bin/systemctl
 status nginx' ,START_DEPENDENCIES='hard(ora.net1.network)
 pullup(ora.net1.network)',
STOP_DEPENDENCIES='hard(intermediate:ora.net1.network)',
 RESTART_ATTEMPTS=0, HOSTING_MEMBERS='dc1north01,dc1north02', CARDINALITY=2"
```

The NGINX resource created in this example run on the named cluster nodes at the same time, specified by `HOSTING_MEMBERS`. This is recommended when multiple GoldenGate Service Manager deployments are configured, and they can independently move between cluster nodes.

Once the NGINX Clusterware resource is created, alter the GoldenGate XAG resources so that NGINX must be started before the GoldenGate deployments are started.

As the `oracle` user, modify the XAG resources using the following example commands.

Determine the current `--filesystems` parameter:

```
$ agctl config goldengate SOURCE|grep "File System"

File System resources needed: ora.datac1.acfs_gg.acfs
```

Modify the `--filesystems` parameter:

```
$ agctl modify goldengate SOURCE --filesystems ora.datac1.acfs_gg.acfs,nginx
```

Repeat the above commands for each of the XAG GoldenGate registrations relying on NGINX.

Task 9: Create Oracle Net TNS Alias for Oracle GoldenGate Database Connections

Create a TNS alias on **all** of the Oracle RAC nodes where Oracle GoldenGate may be started to provide local database connections for the GoldenGate processes when switching between Oracle RAC nodes. Create the TNS alias in the `tnsnames.ora` file in the `TNS_ADMIN` directory specified in the deployment creation.

If the source database is a multitenant database, two TNS alias entries are required: one for the container database (CDB) and one for the pluggable database (PDB) that is being replicated. For a target multitenant database, the TNS alias connects the PDB where replicated data is being applied to. The pluggable database `SERVICE_NAME` should be set to the database service created in an earlier step (refer to Task 3: Create the Database Services).

Below are some example source database TNS alias definitions using the IPC protocol, which must be defined locally on all RAC nodes.

```
OGGSOURCE_CDB =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL=IPC)(KEY=LISTENER))
      (CONNECT_DATA =
        (SERVICE_NAME = oggserv_cdb)
       )
    )

OGGSOURCE_PDB =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL=IPC)(KEY=LISTENER))
      (CONNECT_DATA =

        (SERVICE_NAME = oggserv_pdb)
       )
    )
```

> **Note:**
>
> When the `tnsnames.ora` or `sqlnet.ora`, located in the `TNS_ADMIN` directory for the GoldenGate deployment, are modified, the deployment needs to be restarted in order to pick up the changes.

With the GoldenGate deployment created, use the Administration Server home page to create the database credentials using the above TNS alias names. See Figure 6 below for an example of the database credential creation using the TNS alias appended to the database user name in the ‘User ID” field.

If the source database is a multitenant database, create database credentials for the CDB and PDB. If the target database is a multitenant database, create a single credential for the PDB.

Task 10: Configure Oracle GoldenGate Processes

When creating Extract, Distribution Paths, and Replicat processes with Oracle GoldenGate Microservices Architecture, all files that need to be shared between Oracle RAC nodes are already shared with the deployment files stored on a shared file system (DBFS or ACFS).

Listed below are important configuration details that are recommended for running Oracle GoldenGate Microservices on Oracle RAC for Extract, Distribution Paths and Replicat processes.

Extract Configuration

1. When creating an Extract using the Oracle GoldenGate Administration Server GUI interface, leave the **Trail SubDirectory** parameter blank, so that the trail files are automatically created in the deployment directories stored on the shared file system.

   The default location for trail files is the `/<deployment directory>/var/lib/data`

2. If you are using DBFS for shared storage, and the deployment `var/temp` directory was moved to local storage as described in Task 6: Create the Oracle GoldenGate Deployment, it is recommended that you use the Extract `CACHEMGR` parameter to place the temporary cache files on the shared storage.

Create a new directory under the DBFS deployment mount point. For example:

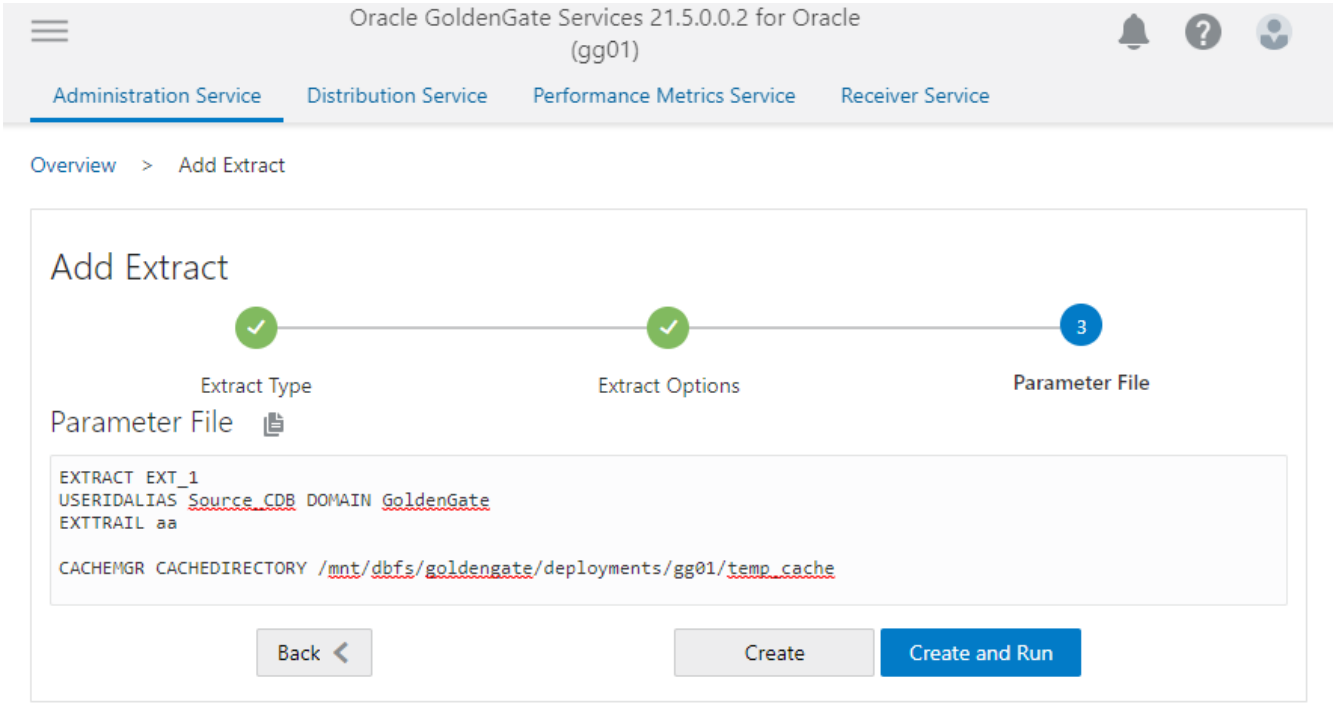
```
$ mkdir -p /mnt/dbfs/goldengate/deployments/ggnorth/temp_cache
```

Set the Extract parameter to the new directory:

```
CACHEMGR CACHEDIRECTORY /mnt/dbfs/goldengate/deployments/ggnorth/temp_cache
```

Shown below is an example of how the parameters specified for an integrated Extract with the Oracle GoldenGate Administration Server GUI looks in the UI.

Figure 24-1 Extract parameters for defining the temporary cache files



Distribution Path Configuration

When using Oracle GoldenGate distribution paths with the NGINX Reverse Proxy, there are additional steps that must be performed to ensure that the path server certificates are configured.

Follow the instructions provided in the following video to correctly configure the certificates: https://apexapps.oracle.com/pls/apex/f?p=44785:112:0::::P112_CONTENT_ID:31380

Configuration highlights presented in this video:

1. Create a client certificate for the source deployment and add the client certificate to the source deployment Service Manager. (This is not required when using Oracle GoldenGate 21c or later releases.)

2. Download the target deployment server's root certificate and add the CA certificate to the source deployment Service Manager.
3. Create a user in the target deployment for the distribution path to connect to.
4. Create a credential in the source deployment connecting to the target deployment with the user created in the previous step.

   For example, a domain of `GGNORTH_to_GGSOUTH` and an alias of `PathReceiver`.

After configuring the client and server certificates, the following configuration options need to be set. Refer to the figures below to see where these options are set in the UI.

1. Change the **Generated Source URI** specifying `localhost` for the server name.

   This allows the distribution path to be started on any of the Oracle RAC nodes.
2. Set the **Target Authentication Method** to `UserID Alias` and the **Target** transfer protocol to `wss` (secure web socket).

   Set the **Target Host** to the target host name/VIP that will be used for connecting to the target system along with the **Port Number** that NGINX was configured with (default is 443).

   The target host name/VIP should match the common name in the CA signed certificate used by NGINX.
3. Set the **Domain** to the credential domain created above in step 4 and presented in the video, for example `GGNORTH_to_GGSOUTH`.

   The **Alias** is set to the credential alias, also created in step 4 in the video.
4. Set the distribution path to automatically restart when the Distribution Server starts.

   This is required so that manual intervention is not required after an Oracle RAC node relocation of the Distribution Server. It is recommended that you set the number of **Retries** to 10. Set the **Delay**, which is the amount of time in minutes to pause between restart attempts, to 1.

Figure 24-2 Distribution Path Creation steps 1-3

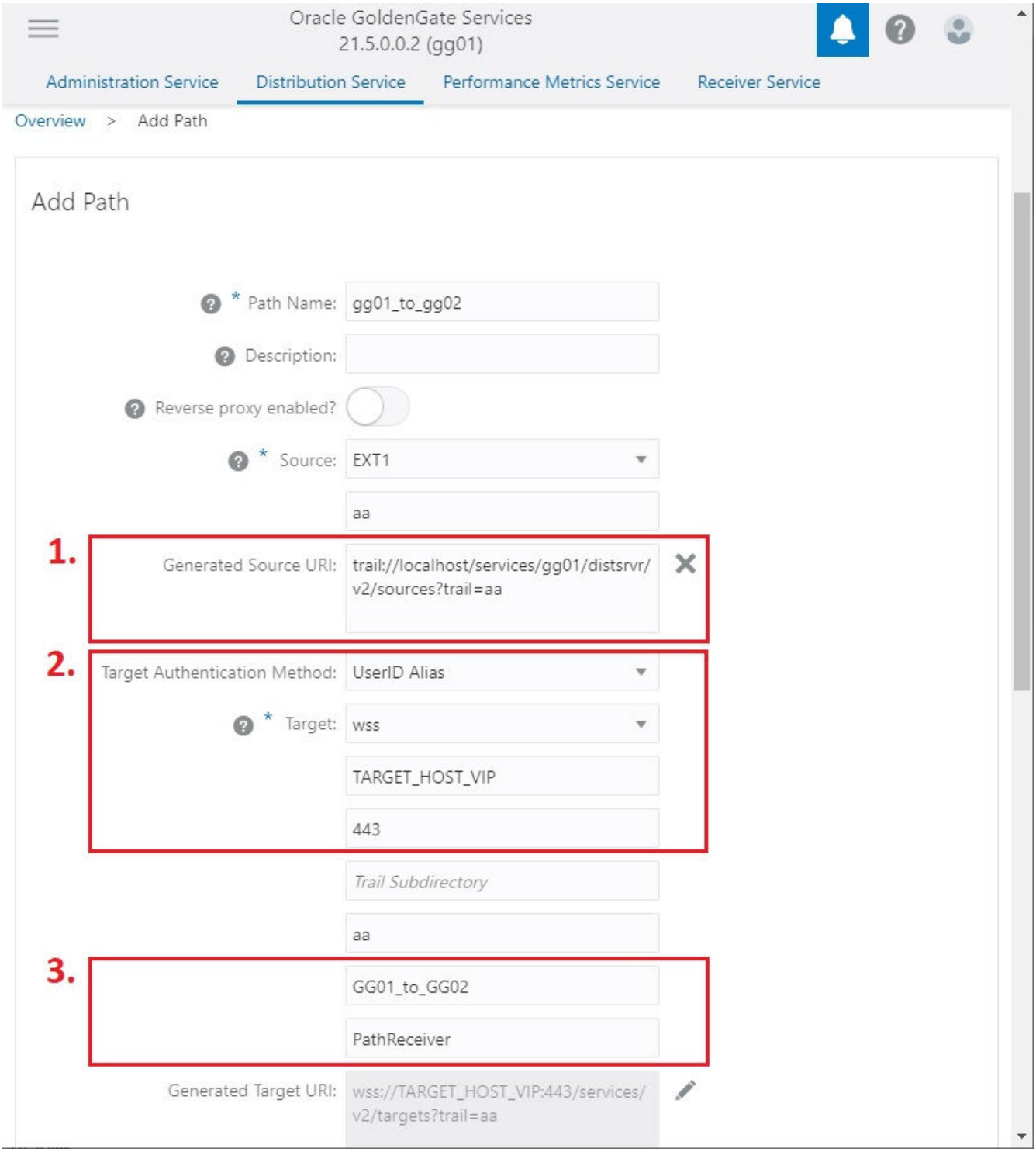


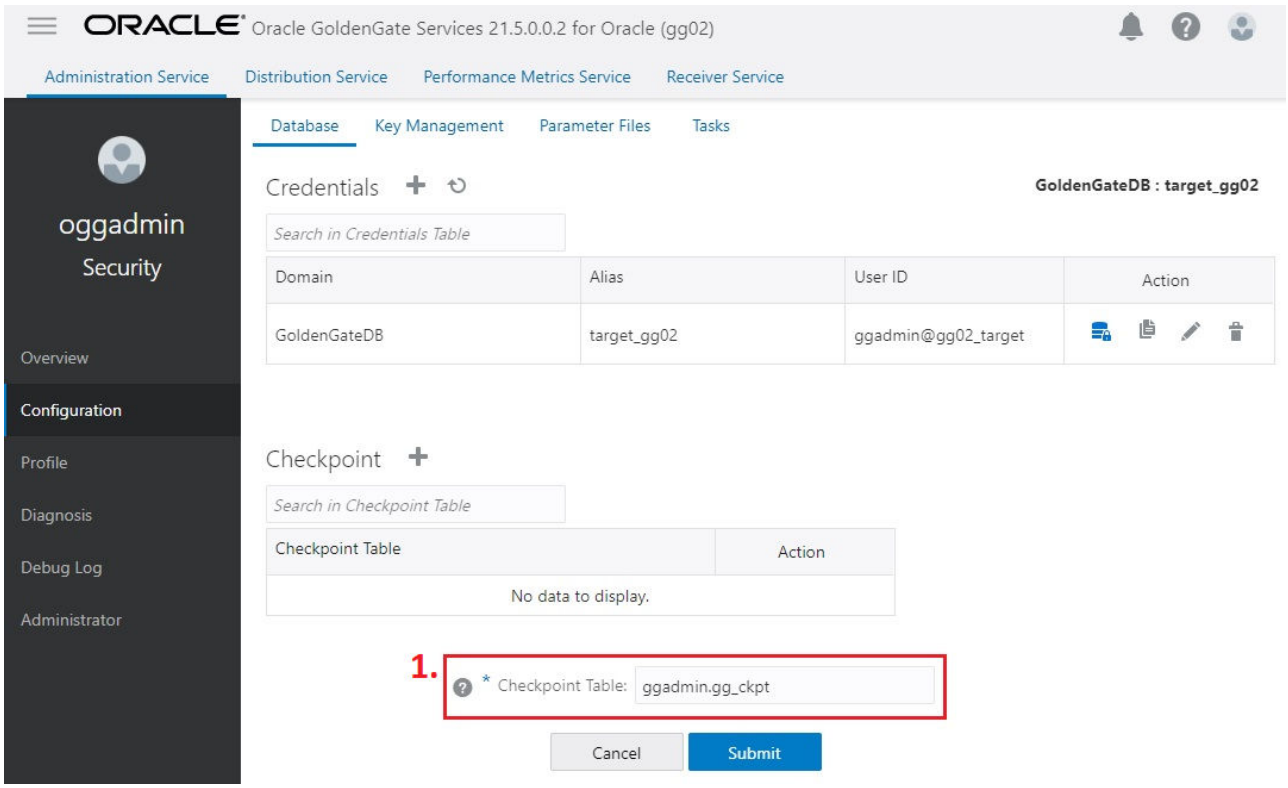
Figure 24-3 Distribution Path Creation step 4

Replicat Configuration

1. The checkpoint table is a required component for GoldenGate Replicat processes. Make sure that a checkpoint table has been created in the database GoldenGate administrator (GGADMIN) schema.

   The checkpoint table can be created using the Oracle GoldenGate Administration Server GUI, clicking on the '+' button and entering the checkpoint table name in the format of schema.tablename. This is shown in the image below

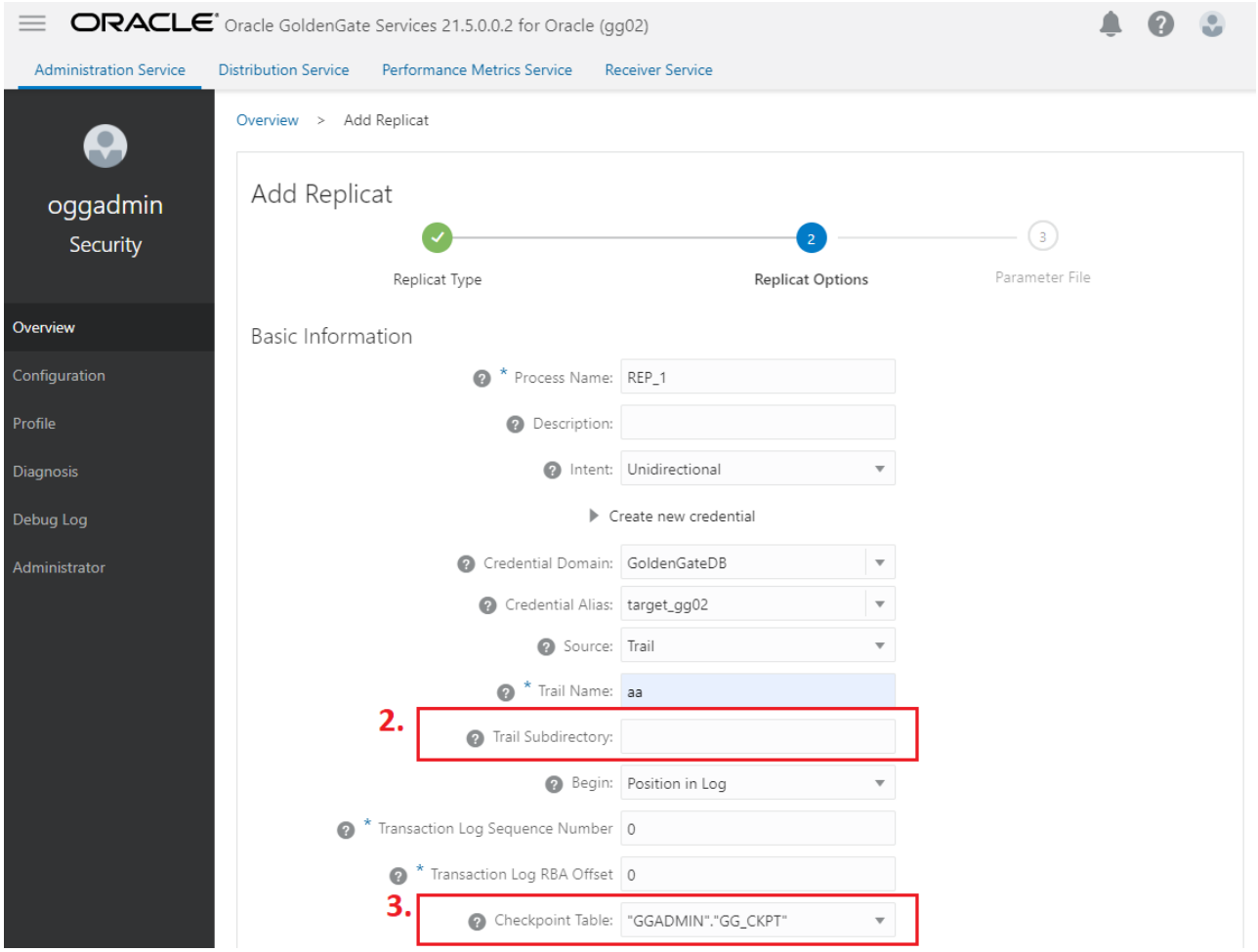
Figure 24-4 Creating the checkpoint table for Replicat processes



   See About Checkpoint Table for more information about creating a checkpoint table.

2. When creating a Replicat using the Oracle GoldenGate Administration Server GUI interface, set the **Trail SubDirectory** parameter to the location where the distribution path or local Extract are creating the trail files.
3. If a checkpoint table was created previously, select the table name from the **Checkpoint Table** pulldown list.

Figure 24-5 Replicat creation with Trail SubDirectory and Checkpoint Table



Task 11: Configure Autostart of Extract and Replicat Processes

Configure the Extract and Replicat processes to automatically start when the Oracle GoldenGate Administration Server is started, and then to restart if any Extract or Replicat processes abend. With GoldenGate Microservices auto start and restart is managed by Profiles.

Using the Oracle GoldenGate Administration Server GUI, create a new profile which can be assigned to each of the Oracle GoldenGate processes.

Profile Configuration Option	Recommended Setting
Default Profile	Enabled
Auto Start	Enabled
Startup Delay	1 minute
Auto Restart	Enabled
Max Retries	5
Retry Delay	30 seconds
Retries Window	30 minutes
Restart on Failure only	Enabled

Profile Configuration Option	Recommended Setting
Disable Task After Retries Exhausted	Enabled

After the profile has been created, and set as the default profile, all new GoldenGate processes created are assigned this profile. For all existing processes, the profile must be assigned to each process.

In the Overview pane, on the Process Information tab, select the **Profile Name** under **Managed Options**.

> **Note:**
>
> When using Oracle GoldenGate Microservices with XAG, it is strongly recommended not to enable the 'Critical to deployment health' flag for any Extract or Replicat processes. Doing so can cause an entire GoldenGate deployment outage from a single Extract or Replicat failure, and also prevents XAG from being able to restart GoldenGate. Refer to Troubleshooting Oracle GoldenGate on Oracle RAC for an example of troubleshooting an outage caused by setting a Replicat to critical.

25

On-Premises MAA Platinum: Oracle GoldenGate Microservices Architecture Integrated with Active Data Guard

The combination and integration of Oracle GoldenGate Microservices and Oracle Data Guard enables you to achieve an MAA Platinum service-level configuration that achieves zero or near zero downtime for all planned and unplanned outages.

Follow these configuration best practices to enable Oracle GoldenGate Microservices replication using a database that is protected by a Data Guard standby, to transparently and seamlessly work following an Oracle Data Guard role transition, no matter which Data Guard protection mode is configured (Maximum Performance, Maximum Availability, or Maximum Protection).

Topics:

Prerequisites

Be sure to complete the following prerequisites before performing any tasks for on-premises MAA Platinum architecture configuration.

- As a prerequisite for MAA Platinum on-premises, have Oracle GoldenGate configured as detailed in On-Premises: Oracle GoldenGate Microservices Architecture with Oracle Real Application Clusters Configuration Best Practices.
- The Database File System (DBFS) is required for critical Oracle GoldenGate files when integrating with Data Guard.
- The Oracle Data Guard standby database should also be configured and operational before continuing.

The following are software requirements that the MAA Platinum configuration is based on:

- Oracle Grid Infrastructure 19c or later

  Oracle Grid Infrastructure provides the necessary components needed to manage high availability for any business-critical applications. Using Oracle Clusterware (a component of Oracle Grid Infrastructure) network, database, and Oracle GoldenGate resources can be managed to provide availability in the event of a failure.

- Oracle Grid Infrastructure Agent version 10.2 or later

  The Oracle Grid Infrastructure Agent leverages the Oracle Grid Infrastructure components to provide integration between Oracle GoldenGate and its dependent resources, such as the database, network, and file system. The agent also integrates Oracle GoldenGate with Oracle Data Guard so that Oracle GoldenGate is restarted on the new primary database following a role transition.

- Oracle Database 19c or later

  See My Oracle Support Document 2193391.1 for a full list of recommended Oracle Database patches when using Oracle GoldenGate.

- Oracle GoldenGate Microservices version 21c or later

  Oracle GoldenGate 21c introduces unified build support so a single software installation supports capturing and applying replicated data to multiple major Oracle Database versions (11g Release 2 to 21c). This is possible because an Oracle GoldenGate installation includes the required Oracle Database client libraries without requiring a separate database `ORACLE_HOME` installation.

- Oracle DBFS to protect and replicate critical Oracle GoldenGate files

  The Oracle Database File System (DBFS) is the only MAA-validated and recommended file system for an Oracle Data Guard and Oracle GoldenGate configuration, because it allows the storage of the required Oracle GoldenGate files, such as the checkpoint and trail files, to be located inside the same database that is protected with Oracle Data Guard, ensuring consistency between the Oracle GoldenGate files and the database in a seamless fashion.

When the prerequisites are met, follow the configuration best practices in the Tasks that follow. These tasks should be performed to ensure the seamless integration of Oracle GoldenGate Microservices with Oracle Data Guard, which in turn ensures that GoldenGate continues running after any Data Guard role transition.

Task 1: Configure the Standby Database for Oracle GoldenGate

The standby database initialization parameters should match those of the primary database.

See Task 1: Configure the Oracle Database for Oracle GoldenGate for details. This includes the following parameters:

- `ENABLE_GOLDENGATE_REPLICATION=TRUE`
- For Oracle GoldenGate source databases, enable `FORCE LOGGING` mode and enable minimal supplemental logging.
- If a GoldenGate source database, or running integrated Replicat (parallel or non-parallel), configure the `STREAMS_POOL_SIZE`.

Task 2: Modify the Primary Database Service

On the primary database server, modify the existing database service that was created as part of the original Oracle GoldenGate on Oracle RAC configuration.

Set the service role to `PRIMARY`, so that the service is only be started when the database becomes the Data Guard primary database role after a role transition.

As the `oracle` user, modify the service using the following command:

```
$ srvctl modify service -db dbName -service service_name
 -role PRIMARY
```

If your database is part of a multitenant environment, remember to modify both the multitenant container database (CDB) and pluggable database (PDB) services.

Task 3: Create the Standby Database Service

On the standby cluster, a database service is required for the standby database so that the Oracle Grid Infrastructure Agent automatically starts the Oracle GoldenGate deployment when the database is opened with the primary role.

When a source database is in a multitenant environment, a separate service is required for the root container database (CDB) and the pluggable database (PDB) that contains the schema being replicated. For a multitenant environment target database, a single service is required for the PDB.

Create the service using the following command, as the `oracle` user, the same way the service was created on the primary cluster.

```
$ srvctl add service -db dbName -service service_name
 -preferred instance_1 -available instance_2, instance_3 etc.
 -pdb pdbName -role PRIMARY
```

It is recommended that you use the same service name as was specified on the primary cluster. The service must be created as a singleton service, using the `-preferred` option, because the application Virtual IP address (VIP), DBFS, and Oracle GoldenGate run on the cluster node where the service is running.

If the database is not in a multitenant environment, or the database is a target database for Oracle GoldenGate, omit the `-pdb` parameter.

Task 4: Configure DBFS on the Standby Cluster Nodes

The Database File System (DBFS) is the only recommended solution when configuring Oracle GoldenGate with Oracle Data Guard.

The DBFS user, tablespace, and file system in the database was previously created in the primary database, as detailed in Task 4: Set Up a File System on Oracle RAC.

The remaining configuration steps are required on all nodes of the standby cluster where Oracle GoldenGate may run.

1. Install the required FUSE libraries, if they are not already installed, by following the instructions in My Oracle Support Document 869822.1.
2. Create the `tnsnames.ora` Oracle Net connection alias using the IPC protocol, similar to the one created on the primary cluster.

```
dbfs =
     (DESCRIPTION =
       (ADDRESS = (PROTOCOL = IPC)(KEY=LISTENER))
       (CONNECT_DATA =
         (SERVICE_NAME = NAME)
        )
     )
```

3. Create the **same** mount point for DBFS that is used on the primary cluster.

   It is important that the mount point is identical, because the physical location of the Oracle GoldenGate deployment is included in the deployment configuration files.

   For example:

```
# mkdir /mnt/dbfs
```

4. Copy the `mount-dbfs.conf` and `mount-dbfs.sh` files from the primary cluster to the standby cluster nodes.

   It is recommended that you place them in the same directory as the primary cluster.
5. Register the DBFS resource with Oracle Clusterware, using the following example command.

   If you are using Oracle Multitenant, make sure to use the service name for the same PDB that contains the DBFS repository as was created in the primary database.

```
DBNAME=dbName
DEPNAME=ora.$DBNAME.oggserv_pdb.svc

crsctl add resource $RESNAME \
  -type cluster_resource \
  -attr "ACTION_SCRIPT=$ACTION_SCRIPT, \
         CHECK_INTERVAL=30,RESTART_ATTEMPTS=10, \
         START_DEPENDENCIES='hard($DEPNAME)pullup($DEPNAME)',\
         STOP_DEPENDENCIES='hard($DEPNAME)',\
         SCRIPT_TIMEOUT=300"
```

Task 5: Install Oracle GoldenGate Software

Install the Oracle GoldenGate software locally on all nodes in the standby cluster that will be part of the Oracle GoldenGate configuration.

Make sure the installation directory is **identical** on all nodes to match the primary cluster installation directory.

Download the Oracle GoldenGate 21c software, or later version, at this location:

http://www.oracle.com/technetwork/middleware/goldengate/downloads/index.html

Task 6: Create Oracle GoldenGate Deployment Directories

The Oracle GoldenGate Service Manager and deployment are already created on the primary cluster, as required by the prerequisites, but certain directories and symbolic links need to be configured on the standby cluster nodes.

These directories and symbolic links were created on the primary cluster, in the tasks you performed as part of On-Premises: Oracle GoldenGate Microservices Architecture with Oracle Real Application Clusters Configuration Best Practices.

Now you create the following directories and symbolic links on the all Oracle RAC nodes on the standby cluster as follows.

1. If there are multiple GoldenGate Service Managers configured on the primary cluster, each with their own deployment, and individually registered with XAG, they must belong to separate `OGG_HOME` software installation directories.

   The same directories and symbolic links for the `OGG_HOME` directories that were configured on primary cluster, must match on the standby cluster.

2. If the GoldenGate deployment was created with the Performance Metric Server enabled, the metric datastore home directory must be created on the standby Oracle RAC nodes.

   For example, determine the datastore directory on the primary cluster nodes:

   ```
   $ grep RepoDatastorePath <deployment directory>/var/log/pmsrvr.log|uniq

   "RepoDatastorePath": "",
    "RepoDatastorePath": "/u01/oracle/goldengate/datastores/ggnorth",
   ```

   Then create the directory on all standby cluster nodes:

   ```
   $ mkdir -p /u01/oracle/goldengate/datastores/ggnorth
   ```

3. If the database release is earlier than Oracle Database 21c (21.3), create the Oracle GoldenGate deployment temp directory local storage to match the symbolic link created on the primary cluster.

   For example, on the primary cluster if you have:

   ```
   $ ls -lrt DBFS_GoldenGate_deployment_home_directory/var/temp

   lrwxrwxrwx  1 oracle oinstall 32 Aug 31 12:27 temp
    -> /u01/oracle/goldengate/deployments/ggnorth/temp
   ```

   Then create the same directory on the standby cluster nodes:

   ```
   $ mkdir -p /u01/oracle/goldengate/deployments/ggnorth/temp
   ```

Task 7: Configure the Standby NGINX Reverse Proxy

Follow these steps to configure the standby NGINX reverse proxy.

1. Install NGINX Reverse Proxy.

If NGINX Reverse Proxy has not already been installed, follow the installation instructions at https://nginx.org/en/linux_packages.html.

As the `root` user, copy the Oracle GoldenGate deployment NGINX configuration files from a primary cluster node to a single standby node directory `/etc/nginx/conf.d`.

For example:

```
[root@dc2north01]# scp dc1north01:/etc/nginx/conf.d/ogg_north.conf
 /etc/nginx/conf.d
```

The standby cluster will need a different CA signed certificate due to using a different VIP name/address than the primary cluster. Contact your systems administrator to follow your corporate standards to create or obtain the server certificate before proceeding. A separate certificate is required for each VIP and Service Manager pair.

2. Install server certificates for NGINX.

   Install the server CA certificates and key files in the `/etc/nginx/ssl` directory, owned by `root` with file permissions 400 (-r--------):

   ```
   # mkdir /etc/nginx/ssl
   # chmod 400 /etc/nginx/ssl
   ```

   For each reverse proxy configuration file copied from the primary cluster, set the correct file names for the certificate and key file using the following example:

   ```
   ssl_certificate /etc/nginx/ssl/gg-stby-vip1.pem;
   ssl_certificate_key /etc/nginx/ssl/gg-stby-vip1.key;
   ```

   When using CA signed certificates, the certificate named with the `ssl_certificate` NGINX parameter must include the root, intermediate, and CA signed certificates in a single file. The order is very important, otherwise NGINX fails to start and displays the error message

   `(SSL: error:0B080074:x509 certificate routines: X509_check_private_key:key values mismatch)`.

   The root and intermediate certificates can be downloaded from the CA signed certificate provider.

   The single file can be generated using the following example command:

   ```
   # cat CA_signed_cert.crt intermediate.crt root.crt
    > gg-stby-vip1.pem
   ```

   The `ssl_certificate_key` file is the key file generated when creating the Certificate Signing Request (CSR), which is required when requesting a CA signed certificate.

   Change the `server_name` parameter in the reverse proxy configuration file copied from the primary cluster, setting to the correct VIP name. For example:

   Before:

   server_name **dc1north**-vip1.example.com;

   After:

   server_name **dc2north**-vip1.example.com;

3. Validate and restart NGINX.

   Because the VIP will not be running on the standby cluster until the primary database service is running, there is a parameter that needs to be set in the `/etc/sysctl.conf` file.

   a. As the `root` user, make the following modifications to `/etc/sysctl.conf`.

      ```
      # vi /etc/sysctl.conf
      ```

   b. Add the following parameter:

      ```
      # allow processes to bind to the non-local address

      net.ipv4.ip_nonlocal_bind = 1
      ```

   c. Reload the modified configuration:

      ```
      # sysctl -p /etc/sysctl.conf
      ```

   d. Validate the NGINX configuration file to detect any errors in the configuration. If there are errors in the file, they will be reported by the following command.

      ```
      # nginx -t
      nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
      nginx: configuration file /etc/nginx/nginxconf test is successful
      ```

   e. Restart NGINX with the new configuration:

      ```
      # systemctl restart nginx
      ```

   When the NGINX configuration is complete, copy the configuration file and certificates to matching directories on the other standby cluster nodes.

4. Create an NGINX Clusterware resource.

   Oracle Clusterware needs to have control over starting the NGINX reverse proxy so that it can be started automatically before the GoldenGate deployments are started.

   The NGINX resource is created with a dependency on the underlying network CRS resource, the name of which can be determined using the following command:

   ```
   $ $GRID_HOME/bin/crsctl stat res -w "TYPE == ora.network.type"|grep NAME
   NAME=ora.net1.network
   ```

   a. As the `root` user, use the following example command to create a Clusterware resource to manage NGINX.

      ```
      # $GRID_HOME/bin/crsctl add resource nginx -type generic_application
       -attr "ACL='owner:root:rwx,pgrp:root:rwx,other::r--,group:oinstall:r-
      x,user:oracle:rwx',EXECUTABLE_NAMES=nginx,START_PROGRAM='/bin/systemctl
       start -f nginx',STOP_PROGRAM='/bin/systemctl
       stop -f nginx',CHECK_PROGRAMS='/bin/systemctl
       status nginx' ,START_DEPENDENCIES='hard(ora.net1.network)
      pullup(ora.net1.network)',
       STOP_DEPENDENCIES='hard(intermediate:ora.net1.network)',
      ```

```
 RESTART_ATTEMPTS=0, HOSTING_MEMBERS='dc1north01,dc1north02',
CARDINALITY=2"
```

The NGINX resource created in this example runs on the named cluster nodes at the same time, specified by `HOSTING_MEMBERS`. This is recommended when multiple GoldenGate Service Manager deployments are configured, and they can independently move between cluster nodes.

b. When the NGINX Clusterware resource is created, alter the GoldenGate XAG resources so that NGINX must be started before the GoldenGate deployments are started.

As the `root` user, modify the XAG resources using the following example commands.

Determine the current `--filesystems` parameter:

```
# agctl config goldengate GGNORTH | grep "File System"

File System resources needed: dbfsgg
```

Task 8: Configure Oracle Clusterware

1. Modify the primary cluster XAG GoldenGate instance.

   The Oracle Grid Infrastructure Standalone Agent (XAG) GoldenGate instance on the primary cluster must be modified as the root user, to identify that it is part of an Oracle Data Guard configuration using the following example command.

   ```
   # agctl modify goldengate instance_name --dataguard_autostart yes
   ```

2. On the standby cluster, follow the instructions in Task 7: Oracle Clusterware Configuration to do steps 3-5 below.
3. Install the XAG software on each standby cluster node.

   It is recommended that you install the XAG software into the same directory as the primary cluster.

4. Prepare for the XAG application VIP creation.

   It is assumed that the VIP and VIP name will be different from that of the primary cluster, so the VIP address will need to be allocated by your systems administrator for the standby cluster.

5. Register Oracle GoldenGate Microservices with XAG.

   The parameters used to register Oracle GoldenGate Microservices with XAG are similar to those used when registering with the primary cluster.

   a. Determine the current parameters in the primary cluster using the following command:

   ```
   $ agctl config goldengate GoldenGate_instance_name

   Instance name: GoldenGate_instance_name
   Application GoldenGate location is: /u01/oracle/goldengate/gg21c_MS
   Goldengate MicroServices Architecture environment: yes
   Goldengate Service Manager configuration directory:
    /mnt/dbfs/goldengate/deployments/ggnorth_sm/etc/conf
   ```

```
Goldengate Service Manager var directory:
 /mnt/dbfs/goldengate//deployments/ggnorth_sm/var
Service Manager Port: 9100
Goldengate Administration User: oggadmin
Autostart on DataGuard role transition to PRIMARY: yes
Configured to run on Nodes: dc1north01,dc1north02
ORACLE_HOME location is: /u01/oracle/goldengate/gg21c_MS/lib/
instantclient
Database Services needed:
ora.ggdg.oggserv_cdb.svc,ora.ggdg.oggserv_pdb.svc
File System resources needed: dbfsgg,nginx
VIP name: gg_vip_prmy
```

In addition, the XAG parameter `--filesystem_verify no` must be specified to prevent XAG from checking the existence of the DBFS deployment directory when registering the GoldenGate instance. Without setting this parameter, the XAG registration will fail, because DBFS is not mounted on the standby cluster.

> **Note:**
>
> It is recommended that you use the same GoldenGate instance name when registering GoldenGate with XAG as was used in the primary cluster.

b. Register GoldenGate with XAG on the standby cluster, as the `root` user:

```
# agctl add goldengate GoldenGate_instance_name \
--gg_home /u01/oracle/goldengate/gg21c_MS \
--service_manager \
--config_home /mnt/dbfs/goldengate/deployments/ggnorth_sm/etc/conf \
--var_home /mnt/dbfs/goldengate/deployments/ggnorth_sm/var \
--port 9100 \
--oracle_home /u01/goldengate/gg21c_MS/lib/instantclient \
--adminuser oggadmin \
--user oracle \
--group oinstall \
--vip_name gg_vip_stby \
--filesystems dbfsgg,nginx \
--db_services ora.ggdgs.oggserv_cdb.svc,ora.ggdgs.oggserv_pdb.svc \
--use_local_services \
--nodes dc2north01,dc2north02 \
--filesystem_verify no \
--dataguard_autostart yes
```

For more information about the Oracle Grid Infrastructure Bundled Agent, see http://www.oracle.com/technetwork/database/database-technologies/clusterware/downloads/xag-agents-downloads-3636484.html

Task 9: Create Oracle Net TNS Aliases for Oracle GoldenGate Database Connections

The same TNS aliases created on the primary cluster for the primary database using the IPC protocol must be created with the **same** alias names on each node of the standby cluster, using the IPC communication protocol as specified in Task 9: Create Oracle Net TNS Alias for Oracle GoldenGate Database Connections.

The location of `tnsnames.ora` used by the Oracle GoldenGate deployment **must** be the **same** on the standby cluster nodes as it is on the primary cluster.

Use the following query REST API call to query the `TNS_ADMIN` location on the primary cluster.

```
$ curl -s -u OGG_admin_username
 https://vip_name/services/v2/deployments/deployment_name
 -XGET|python -m json.tool|grep TNS_ADMIN -A1
```

You will be prompted to enter the Oracle GoldenGate Service Manager administrator user password.

For example:

```
$ curl -s -u oggadmin https://dc1north01-vip1/services/v2/deployments/ggnorth
 -XGET|python -m json.tool|grep TNS_ADMIN -A1

                "name": "TNS_ADMIN",
                "value": "/u01/goldengate/network/admin"
```

Make sure the `tnsnames.ora` is located in this **same** directory on **all** standby cluster nodes.

Example TNS alias for the GoldenGate database:

```
ggnorth_pdb =
    (DESCRIPTION =
      (SDU = 2097152)
      (ADDRESS = (PROTOCOL = IPC)(KEY=LISTENER))
      (CONNECT_DATA =
        (SERVICE_NAME = oggserv_pdb.example.com)
       )
    )
```

Task 10: Configure Oracle GoldenGate Processes

In addition to the guidance provided in Task 10: Configure Oracle GoldenGate Processes, follow the recommendations provided below for Extract, Distribution Paths, and Replicats.

Extract Configuration on the Primary Cluster

For GoldenGate Extract processes using Data Guard configurations that are using redo transport Maximum Performance or Maximum Availability modes, the following parameter must

be added to the Extract process parameter file **on the primary cluster** to avoid losing transactions and resulting in logical data inconsistencies:

```
TRANLOGOPTIONS HANDLEDLFAILOVER
```

This parameter prevents Extract from extracting transaction data from redo that has not yet been applied to the Data Guard standby database. This is crucial to preventing Oracle GoldenGate from replicating data to a target database that does not exist in the source standby database.

If this parameter is not specified, after a data loss failover of the source database it is possible to have data in the target database that is not present in the source database, leading to logical data inconsistencies.

By default, after 60 seconds, a warning message will be written to the Extract report file when the Extract is stalled due to not being able to query the standby database applied SCN information. For example:

```
WARNING OGG-02721 Extract has been waiting for the standby database for 60
seconds.
```

The amount of time before the warning message is written to Extract report file can be adjusted using the Extract parameter `TRANLOGOPTIONS HANDLEDLFAILOVER STANDBY_WARNING`.

If the Extract is still not able to query the standby database applied SCN information after 30 minutes (default), the Extract process will abend, logging the following message in the Extract report file:

```
ERROR OGG-02722 Extract abended waiting for 1,800 seconds for the standby
database to be accessible or caught up with the primary database.
```

If the standby database becomes available before the 30 default timeout expires, Extract continues mining data from the source database and reports the following message to the report file:

```
INFO OGG-02723 Extract resumed from stalled state and started processing LCRs.
```

The timeout value of 30 minutes can be adjusted using the Extract parameter `TRANLOGOPTIONS HANDLEDLFAILOVER STANDBY_ABEND` *`value`*, where *value* is the number of seconds the standby is unavailable before abending.

If the standby database will be unavailable for a prolonged duration, such as during a planned maintenance outage, and you wish Extract to continue extracting data from the primary database, remove the `TRANLOGOPTIONS HANDLEDLFAILOVER` parameter from the Extract parameter file and restart Extract. Remember to set the parameter after the standby becomes available.

> **Note:**
>
> If extracting from a primary database continues while the standby is unavailable, a data loss failover could result after the standby becomes available, and not all the primary redo was applied before a failover. The GoldenGate target database will contain data that does not exist in the source database.

See *Oracle GoldenGate Reference Guide* for more information about the `TRANLOGOPTIONS HANDLEDLFAILOVER` parameters at https://docs.oracle.com/en/middleware/goldengate/core/21.3/reference/reference-oracle-goldengate.pdf.

If the Extract process has been assigned an auto restart profile, as documented in Task 11: Configure Autostart of Extract and Replicat Processes, after a Data Guard role transition, the Extract process will automatically restart. Extract will continue to mine redo data from the new primary database, ignoring the current state of the new standby database, until a default 5 minute timeout period expires. After this time, if the standby is not available Extract will abend with the following errors:

```
INFO OGG-25053 Timeout waiting for 300 seconds for standby database
reinstatement. Now enforcing HANDLEDLFAILOVER.

ERROR OGG-06219 Unable to extract data from the Logmining server OGG$CAP_EXT1.

ERROR OGG-02078 Extract encountered a fatal error in a processing thread and is
abending.
```

Extract will continue to automatically restart, based on the Oracle GoldenGate Microservices auto restart profile, and failing due to reaching the `HANDLEDLFAILOVER` timeout, until the number retries is reached or the new standby database becomes available.

During the timeout period following a database role transition, the `HANDLEDLFAILOVER` parameter is automatically suspended, so data will be replicated to the Oracle GoldenGate replica database without consideration of the source standby database not being kept up to date. The timeout period for the standby database to start up before Extract abends can be adjusted using the Extract parameter `TRANLOGOPTIONS DLFAILOVER_TIMEOUT`.

It is recommended that you leave `DLFAILOVER_TIMEOUT` at the default of 5 minutes, to allow the old primary to convert to a standby. If the new standby database will be unavailable for an extended period of time or completely gone, then in order for Extract to start and remain running, you must remove the `HANDLEDLFAILOVER` parameter from the Extract parameter file. After removing the parameter, Extract no longer waits until redo has been applied to the standby database before extracting the data.

During the time it takes for the standby database to come back online and apply all the redo from the primary

database, there will be data divergence between it and the Oracle GoldenGate replica database. This will be resolved once the standby database is up to date. At which point, add the `HANDLEDLFAILOVER` parameter back into the integrated Extract process parameter file, and then stop and restart the Extract.

When Oracle Data Guard is configured with fast-start failover, such that the broker can automatically fail over to a standby database in the event of loss of the primary database, you must specify an additional integrated Extract parameter shown below.

```
TRANLOGOPTIONS FAILOVERTARGETDESTID n
```

This parameter identifies which standby database the Oracle GoldenGate Extract process must remain behind, with regards to not extracting redo data that has not yet been applied to the standby database.

To determine the correct value for `FAILOVERTARGETDESTID`, use the `LOG_ARCHIVE_DEST_N` parameter from the GoldenGate source database which is used for sending redo to the source standby database. For example, if `LOG_ARCHIVE_DEST_2` points to the standby database, then use a value of 2.

For example:

```
SQL> show parameters log_archive_dest

NAME                 TYPE     VALUE
-------------------- --------
------------------------------------------------
log_archive_dest_1   string   location=USE_DB_RECOVERY_FILE_DEST,
                               valid_for=(ALL_LOGFILES, ALL_ROLES)

log_archive_dest_2   string   service="ggnorths", SYNC AFFIRM delay=0
                               optional compression=disable max_failure=0
reopen=300
                               db_unique_name="GGNORTHS" net_timeout=30,
                               valid_for=(online_logfile,all_roles)
```

In this example, the Extract parameter would be set to the following:

```
TRANLOGOPTIONS FAILOVERTARGETDESTID 2
```

To add the parameters to the Extract parameter file, use the Oracle GoldenGate Administration Server to select display the Extract details

1. "On the **Administration Service** tab, select the **Actions** menu for the Extract and choose **Details."**
2. In the Extract details view select the **Parameters** tab, and then select the pencil icon to edit the current parameter file
3. Add the `TRANLOGOPTIONS` parameters and select **Apply** to save the changes.

For the new parameters to take effect, the Extract process needs to be stopped and restarted, which can be done using the Administration Server.

More information about the Extract `TRANLOGOPTIONS` parameters mentioned above, can be found in the *Reference for Oracle GoldenGate* at https://docs.oracle.com/en/middleware/goldengate/core/21.3/reference/tranlogoptions.html#GUID-B6ADFEC9-10E6-456D-9477-088513E113AF.

Distribution Path Configuration on the Primary and Standby Cluster

When the target database of an Oracle GoldenGate environment, where the Receiver Server runs, is protected with Oracle Data Guard, there is an important consideration that must be given to any Distribution Paths that are sending trail files to the Receiver Server. When the Receiver Server moves to a different cluster after an Oracle Data Guard role transition, any distribution paths must be altered to reflect the new target cluster address.

You can automatically change the distribution paths using a database role transition trigger in the target database on the Receiver Server cluster.

If the primary and standby cluster VIPs use different root CA certificates, the standby certificate will need to be added to the source deployment Service Manager, as detailed in On-Premises: Oracle GoldenGate Microservices Architecture with Oracle Real Application Clusters Configuration Best Practices.

Follow the instructions below to create a database role transition trigger to modify the distribution path target address when the receiver server moves between the primary and standby cluster, during target database Data Guard role transitions.

1. Create a shell script to modify the distribution paths.

Example Distribution Path Target Change Script contains an example shell script that can be used to modify a distribution path target address. Refer to the example script comments for setting appropriate variable values.

The script should be placed in the same local directory on **all** Oracle RAC nodes of the primary **and** standby database clusters. Set the script file permissions to 6751.

For example:

```
$ chmod 6751 /u01/oracle/goldengate/scripts/change_path_target.sh
```

The example shell script uses REST API calls to access the GoldenGate distribution path. In order to make the REST API calls secure, it is recommended that you include the GoldenGate deployment administrator user name and password in a configuration file (`access.cfg`), as shown here.

```
$ cat /u01/oracle/goldengate/scripts/access.cfg

user = "oggadmin:<password>"
```

The `access.cfg` file is also referenced in the database role transition trigger below.

2. Create a `DBMS_SCHEDULER` job.

   Creating a `DBMS_SCHEDULER` job is required to run an operating system shell script from within PL/SQL. Create the scheduler job as a SYSDBA user in the root container database (CDB).

   For example:

```
SQL> exec dbms_scheduler.drop_job('gg_change_path_target');
SQL> exec dbms_scheduler.create_job(job_name=>'gg_change_path_target',
 job_type=>'EXECUTABLE', number_of_arguments => 6,
 job_action=>'/u01/oracle/goldengate/scripts/change_path_target.sh',
 enabled=>FALSE);
```

   To run an external job, you must set the `run_user` and `run_group` parameters in the `$ORACLE_HOME/rdbms/admin/externaljob.ora` file to the Oracle database operating system user and group.

   For example:

```
run_user = oracle
run_group = oinstall
```

   The `extrernaljob.ora` must be configured on **all** Oracle RAC nodes of the primary **and** standby database clusters.

3. Create the database role transition trigger.

   Create a role transition trigger on the GoldenGate target database that will fire when a standby database becomes a primary database, changing the distribution path target address, using the following example.

```
CREATE OR REPLACE TRIGGER gg_change_path
AFTER db_role_change ON DATABASE
declare
```

```
  role varchar2(30);
  hostname varchar2(64);
begin
  select database_role into role from v$database;
  select host_name into hostname from v$instance;


DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('gg_change_path_target',1,'source_pri
mary_cluster_VIP');

DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('gg_change_path_target',2,'source_sta
ndby_cluster_VIP');

DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('gg_change_path_target',4,'dist_path_
name');

DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('gg_change_path_target',5,'deployment
_name');
  DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('gg_change_path_target',6, '<dir/
access.cfg>');

  if role = 'PRIMARY' and hostname like 'primary_target_cluster_name%'
  then


DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('gg_change_path_target',3,'primary_ta
rget_cluster_VIP:443');
  elsif role = 'PRIMARY'
  then

DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('gg_change_path_target',3,'standby_ta
rget_cluster_VIP:443');
  end if;
  DBMS_SCHEDULER.RUN_JOB(job_name=>'gg_change_path_target');
end;
/
```

After creating the database trigger, switch the log file on the primary database to ensure the code is propagated to the standby database using the following command:

```
SQL> alter system switch all logfile;
```

Replicat Configuration on the Primary Cluster

As documented in On-Premises: Oracle GoldenGate Microservices Architecture with Oracle Real Application Clusters Configuration Best Practices, a checkpoint table in the target database is required for all Oracle GoldenGate Replicat processes. There are no other configuration requirements for Replicat when configured with Oracle Data Guard.

Example Distribution Path Target Change Script

The following example script can be used to change a source GoldenGate deployment distribution path target address to reflect the new location of the receiver server after a Data Guard role transition. This example assumes the source GoldenGate deployment is configured

in an MAA architecture with Data Guard, such that the distribution server can relocate between a primary and standby cluster.

```
#!/bin/bash

# change_path_target.sh - changes the target host of a GG Distribution Path
when the target
#                          moves between primary/standby clusters.
# Example usage:
# ./change_path_target.sh <primary source VIP>:443 <standby source VIP>:443
<path target VIP> <path name> <deployment name> <credentials file>

SOURCE1=$1      # PRIMARY Distribution Server VIP
SOURCE2=$2      # STANDBY Distribution Server VIP
TARGET=$3       # Distribution path target VIP
DPATH=$4        # Distribution path name
DEP=$5          # Deployment name
ACCESS=$6       # access.cfg file containing the deployment credentials.
Example contents:
                # user = "oggadmin:<password>"

CONNECT=0

#echo "#${i} - `date`:"
LOGFILE=/tmp/ogg_dpatch_change.txt

result=$(curl -si -K $ACCESS https://$SOURCE1/$DEP/distsrvr/services/v2/
sources/$DPATH -X GET| grep HTTP |  awk '{print $2}')

# Will return NULL of nginx not running, 502 if cannot contact server, 200 if
contact to server good, and others (404) for other bad reasons:

if [[ -z $result || $result -ne 200 ]]; then  # Managed to access the Distr
Server
  echo "`date` - Couldn't contact Distribution Server at $SOURCE1
Deployment $DEP ****" >> $LOGFILE
else # Try the other source host:
  echo "`date` - Got status of Distribution Server at $SOURCE1
Deployment $DEP ***"  >> $LOGFILE
  SOURCE=$SOURCE1
  CONNECT=1
fi

if [ $CONNECT -eq 1 ]; then
# For secure NGINX patch destination (wss)
  PAYLOAD='{"target":{"uri":"wss://'${TARGET}'/services/ggnorth/v2/targets?
trail=bb"}}'
  curl -s -K $ACCESS https://$SOURCE/$DEP/distsrvr/services/v2/sources/$DPATH
-X PATCH --data '{"status": "stopped"}'

# Set new target for path:
  curl -s -K $ACCESS https://$SOURCE/$DEP/distsrvr/services/v2/sources/$DPATH
-X PATCH --data "$PAYLOAD"
  echo "`date` - Set path $DPATH on $SOURCE deployment $DEP:" >> $LOGFILE

  curl -s -K $ACCESS https://$SOURCE/$DEP/distsrvr/services/v2/sources/$DPATH
```

```
-X GET | python -m json.tool | grep uri >> $LOGFILE
  curl -s -K $ACCESS https://$SOURCE/$DEP/distsrvr/services/v2/sources/$DPATH
-X PATCH --data '{"status": "running"}'

exit 0
else
  echo "`date` - ERROR: COULDN'T CHANGE DISTRIBUTION PATH ($DPATH) in
Deployement $DEP at $SOURCE! ***" >> $LOGFILE
fi

# If here, means we couldn't connect to either Distribution Servers
exit 1
```

26

Managing Planned and Unplanned Outages for Oracle GoldenGate Hub

There are a number of considerations that must be taken into account when the hub undergoes a planned or unplanned outage of either the primary or standby file system clusters.

Managing Planned Outages

When there is a requirement to perform planned maintenance on the GGHub, some of the CRS resources should be stopped and disabled to prevent them from restarting, or from causing undesirable results when incorrectly instigating a file system failover, or stopping GoldenGate from running.

Use the following recommendations in the event of a planned outage of the primary or standby hub clusters.

For all planned maintenance events:

- Operating system software or hardware updates and patches
- Oracle Grid Infrastructure interim or diagnostic patches
- Oracle Grid Infrastructure quarterly updates under the Critical Patch Update (CPU) program, or Oracle Grid Infrastructure release upgrades
- GGHub software life cycle, including:
  - Oracle GoldenGate
  - Oracle Grid Infrastructure Agent
  - NGINX

High Availability Solutions with Target Outage Time:

Seconds to minutes where GoldenGate replication is temporarily suspended

Step 1: Software update of idle GGHub node

Step 2: GGHub Node Relocate

Step 3: Software update of the remaining inactive GGHub node

GGHub Node Relocate

As the `grid` OS user on the primary GGHub system, relocate the Oracle GoldenGate Instance:

```
[grid@gghub_prim1 ~]$ agctl status goldengate

Goldengate instance 'gghub' is running on gghub_prim1

[grid@gghub_prim1 ~]$ time agctl relocate goldengate gghub

real    0m43.984s
```

```
user    0m0.156s
sys     0m0.049s
```

As the `grid` OS user on the primary GGHub system, check the status of the Oracle GoldenGate Instance:

```
[grid@gghub_prim1 ~]$ agctl status goldengate

Goldengate instance 'gghub' is running on gghub_prim2
```

GGHub Role Reversal for DR events or to move GGHub in the same region as the target database

GGHub role reversal performs an ACFS role reversal so that the standby becomes the new primary. With both the primary and standby file systems online, the `acfsutil repl failover` command ensures that all outstanding primary file system changes are transferred and applied to the standby before the role reversal completes.

When to use GGHub role reversal:

- To move the GGHub deployment close to the target database for replication performance
- To support site outage
- To support site maintenance

As the `grid` OS user on the current standby GGHub node, create the script to perform the ACFS role reversal:

```
[grid@gghub_stby1]$ export ACFS_MOUNT_POINT=/mnt/acfs_gg1
[grid@gghub_stby1]$ export GG_DEPLOYMENT_NAME=gghub
[grid@gghub_stby1]$ ssh `/sbin/acfsutil repl info -c -v $ACFS_MOUNT_POINT|
grep
 'Primary hostname' | awk '{print $3}'| cut -d "@" -f2`
 "agctl stop goldengate $GG_DEPLOYMENT_NAME"
[grid@gghub_stby1]$ /sbin/acfsutil repl failover $ACFS_MOUNT_POINT
[grid@gghub_stby1]$ agctl start goldengate $GG_DEPLOYMENT_NAME
[grid@gghub_stby1]$ agctl status goldengate $GG_DEPLOYMENT_NAME
Goldengate  instance 'gghub' is running on gghub_stby1
```

Alternatively, as the `grid` OS user on any GGHub node, run the script `acfs_role_reversal.sh` to perform the ACFS role reversal:

```
[grid@gghub_stby1]$ sh /u01/oracle/scripts/acfs_role_reversal.sh
 /mnt/acfs_gg1 gghub

################################################################################
##
ACFS Primary Site: gghub_prim_vip1.frankfurt.goldengate.com
ACFS Standby Site: gghub_stby_vip1.frankfurt.goldengate.com
################################################################################
##
Thu Nov 30 17:28:37 UTC 2023 - Begin Stop  GoldenGate gghub
Thu Nov 30 17:28:38 UTC 2023 - End   Stop  GoldenGate gghub
################################################################################
##
```

```
Thu Nov 30 17:28:38 UTC 2023 - Begin ACFS replication sync /mnt/acfs_gg1
Thu Nov 30 17:28:59 UTC 2023 - End   ACFS replication sync /mnt/acfs_gg1
################################################################################
##
Site:                                Primary
Primary status:                      Running
Status:                              Send Completed
Lag Time:                            00:00:00
Retries made:                        0
Last send started at:                Thu Nov 30 17:28:45 2023
Last send completed at:              Thu Nov 30 17:28:55 2023
################################################################################
##
Site:                                Standby
Last sync time with primary:         Thu Nov 30 17:28:45 2023
Status:                              Receive Completed
Last receive started at:             Thu Nov 30 17:28:46 2023
Last receive completed at:           Thu Nov 30 17:28:52 2023
################################################################################
##
Thu Nov 30 17:29:00 UTC 2023 - Begin Role Reversal
Thu Nov 30 17:30:02 UTC 2023 - End Role Reversal
################################################################################
##
ACFS Primary Site: gghub_stby_vip1.frankfurt.goldengate.com
ACFS Standby Site: gghub_prim_vip1.frankfurt.goldengate.com
################################################################################
##
Site:                                Primary
Primary status:                      Running
Status:                              Send Completed
Lag Time:                            00:00:00
Retries made:                        0
Last send started at:                Thu Nov 30 17:29:45 2023
Last send completed at:              Thu Nov 30 17:29:56 2023
################################################################################
##
Site:                                Standby
Last sync time with primary:         Thu Nov 30 17:29:45 2023
Status:                              Receive Completed
Last receive started at:             Thu Nov 30 17:29:50 2023
Last receive completed at:           Thu Nov 30 17:29:50 2023
################################################################################
##
Thu Nov 30 17:30:03 UTC 2023 - Begin Start GoldenGate gghub
Thu Nov 30 17:30:10 UTC 2023 - End   Start GoldenGate gghub
################################################################################
##
```

Managing Unplanned Outages

Expected Impact with Unplanned Outages

When an unplanned outage occurs on either the primary or standby GGHub clusters, there are some instructions to ensure the continuous operation of GoldenGate. Use the following GGHUB failure use cases to guide you in the event of an unplanned outage of the primary and standby GGHUB systems.

Use case #1 – Standby Hub Failure or Primary GGHub Cannot Communicate with the Standby GGHub

If the primary GGhub cannot communicate with the standby GGhub, the following messages will be output into the primary CRS trace file (crsd_scriptagent_grid.trc) on the active cluster node:

```
2023-06-21 12:06:59.506 :CLSDYNAM:1427187456: [acfs_primary]{1:8532:12141}
[check] Executing action script: /u01/oracle/scripts/acfs_primary.scr[check]
2023-06-21 12:07:05.666 :CLSDYNAM:1427187456: [acfs_primary]{1:8532:12141}
[check] WARNING: STANDBY not accessible (attempt 1 of 3))
2023-06-21 12:07:18.683 :CLSDYNAM:1427187456: [acfs_primary]{1:8532:12141}
[check] WARNING: STANDBY not accessible (attempt 2 of 3))
2023-06-21 12:07:31.751 :CLSDYNAM:1427187456: [acfs_primary]{1:8532:12141}
[check] WARNING: STANDBY not accessible (attempt 3 of 3))
2023-06-21 12:07:31.751 :CLSDYNAM:1427187456: [acfs_primary]{1:8532:12141}
[check] WARNING: Problem with STANDBY file system (error: 222)
```

At this time, the standby file system is no longer receiving the primary file system changes. The primary file system and Oracle GoldenGate will continue to function unimpeded.

Use the following action plan with this scenario.

- Check the standby file system, using the command ‘acfsutil repl util verifystandby /mnt/acfs_gg –v’ to determine why the standby hub is inaccessible.
- After fixing the cause of the communication errors, the standby will automaitically catch up applying the outstanding primary file system changes. The warning messages will no longer be reported into the CRS trace file, being replaced with the following message:

```
2023-06-21 12:15:01.720 :CLSDYNAM:1427187456: [acfs_primary]{1:8532:12141}
[check] SUCCESS: STANDBY file system /mnt/acfs_gg is ONLINE
```

Use case #2 – Primary GGHub Failure or Standby GGHub Cannot Communicate with the Primary GGHub

If the standby GGhub cannot communicate with the primary GGhub, the the following messages will be output into the standby CRS trace file (crsd_scriptagent_grid.trc) on the active cluster node:

```
2023-06-21 12:24:03.823 :CLSDYNAM:4156544768: [acfs_standby]{1:10141:2}
[check] Executing action script: /u01/oracle/scripts/acfs_standby.scr[check]
2023-06-21 12:24:06.928 :CLSDYNAM:4156544768: [acfs_standby]{1:10141:2}
[check] WARNING: PRIMARY not accessible (attempt 1 of 3)
2023-06-21 12:24:19.945 :CLSDYNAM:4156544768: [acfs_standby]{1:10141:2}
[check] WARNING: PRIMARY not accessible (attempt 2 of 3)
2023-06-21 12:24:32.962 :CLSDYNAM:4156544768: [acfs_standby]{1:10141:2}
[check] WARNING: PRIMARY not accessible (attempt 3 of 3)
2023-06-21 12:24:32.962 :CLSDYNAM:4156544768: [acfs_standby]{1:10141:2}
[check] WARNING: Problem with PRIMARY file system (error: 222)
```

At this time, it is unlikely that the standby file system is receiving file system changes from the primary file system.

Use the following action plan with this scenario.

- Check the primary file system, using the command ‘acfsutil repl util verifyprimary /mnt/acfs_gg -v’ to determine why the primary hub is inaccessible.
- If the primary file system cluster is down and cannot be restarted, issue an ACFS failover on the standby GGhub:

```
[grid@gghub_stby1]$ /sbin/acfsutil repl failover /mnt/acfs_gg        #
Specify the correct mount point

[grid@gghub_stby1]$ acfsutil repl info -c -v /mnt/acfs_gg |egrep 'Site:|
Primary status|Background Resources:'

Site:                                 Primary
Primary status:                       Running
Background Resources:                 Active
```

- Run the following commands to prepare the acfs_primary resource to start on the new primary hub, and then restart GoldenGate:

```
[grid@gghub_stby1]$ echo "RESTART" > /mnt/acfs_gg/status/acfs_primary

[grid@gghub_stby1]$ agctl start goldengate <instance_name>              #
Specify the GoldenGate instance name

[grid@gghub_stby1]$ agctl status goldengate

Goldengate  instance '<instance_name>' is running on gghubstby-node1
```

- When the old primary file system comes back online, if connectivity is resumed between the new primary and old primary, the old primary file system will automatically convert to the standby.
- If the old primary file system comes back online, but connectivity cannot be established between the primary and standby file systems the acfs_primary resource will detect that node had crashed, and because connectivity to the standby cannot be confirmed, GoldenGate will not be started. This avoids a ‘split-brain’ where two file systems think they are both the primary because they cannot commnunicate with each other.

Use case #3 – Double Failure Case: Primary GGHub Failure and Standby GGHub Connectivity Failure

If the primary GGhub crashes and communication cannot be established with the standby file system when it comes back online, the following messages will be output into the primary CRS trace file (crsd_scriptagent_grid.trc) on the active cluster node:

```
2023-06-21 17:08:52.621:[acfs_primary]{1:40360:36312} [start] WARNING:
PRIMARY file system /mnt/acfs_gg previously crashed
2023-06-21 17:08:55.678:[acfs_primary]{1:40360:36312} [start] WARNING:
STANDBY not accessible - disabling acfs_primary
```

If an attempt is made to manually restart the primary file system, an additional message will be output into the CRS trace file:

```
2023-06-21 17:25:54.224:[acfs_primary]{1:40360:37687} [start] WARNING:
    PRIMARY /mnt/acfs_gg disabled to prevent split brain
```

Use the following action plan with this scenario.

- Check the standby file system, using the command '*acfsutil repl util verifystandby /mnt/acfs_gg -v*' to determine why the standby hub is inaccessible.
- If communication with the the standby file system can re-established, restart GoldenGate on the primary hub:

  ```
  [grid@gghub_prim1]$ agctl start goldengate <instance_name> # Specify the
  GoldenGate instance name

  [grid@gghub_prim1]$ agctl status goldengate

  Goldengate  instance '<instance_name>' is running on gghub_prim1
  ```

- If communication with the standby file system cannot be re-established, use the following commands to restart GoldenGate on the primary hub:

  ```
  [grid@gghub_prim1]$ echo "RESTART" > /mnt/acfs_gg/status/acfs_primary

  [grid@gghub_prim1]$ agctl start goldengate <instance_name>              #
  Specify the GoldenGate instance name

  [grid@gghub_prim1]$ agctl status goldengate

  Goldengate  instance '<instance_name>' is running on gghub_prim1
  ```

- When communication with the standby file system is restored, ACFS Replication will continue to replicate primary file system changes.

27
Troubleshooting Oracle GoldenGate

Topics:

Troubleshooting MAA GoldenGate Hub

Oracle GoldenGate Extract Failure or Error Conditions Considerations

For Oracle GoldenGate Extract processes using Oracle Data Guard configurations that are using redo transport Maximum Performance or Maximum Availability modes, the following parameter must be added to the Extract process parameter file **on the primary database system** to avoid losing transactions and resulting in logical data inconsistencies:

```
TRANLOGOPTIONS HANDLEDLFAILOVER
```

This parameter prevents Extract from extracting transaction data from redo that has not yet been applied to the Data Guard standby database. This is crucial to preventing Oracle GoldenGate from replicating data to a target database that does not exist in the source standby database.

If this parameter is not specified, after a data loss failover of the source database it is possible to have data in the target database that is not present in the source database, leading to logical data inconsistencies.

By default, after 60 seconds, a warning message is written to the Extract report file when the Extract is stalled because it can't query the standby database applied SCN information. For example:

```
WARNING OGG-02721 Extract has been waiting for the standby database for 60
seconds.
```

The amount of time before the warning message is written to the Extract report file can be adjusted using the Extract parameter `TRANLOGOPTIONS HANDLEDLFAILOVER STANDBY_WARNING`.

If the Extract is still unable to query the standby database applied SCN information after 30 minutes (default), the Extract process abend, logging the following message in the Extract report file:

```
ERROR OGG-02722 Extract abended waiting for 1,800 seconds for the standby
database to be accessible or caught up with the primary database.
```

If the standby database becomes available before the default 30 timeout expires, Extract continues mining data from the source database and reports the following message to the report file:

```
INFO OGG-02723 Extract resumed from stalled state and started processing LCRs.
```

The timeout value of 30 minutes can be adjusted using the Extract parameter `TRANLOGOPTIONS HANDLEDLFAILOVER STANDBY_ABEND <value>`, where value is the number of seconds the standby is unavailable before abending.

If the standby database will be unavailable for a prolonged duration, such as during a planned maintenance outage, and you wish Extract to continue extracting data from the primary database, remove the `TRANLOGOPTIONS HANDLEDLFAILOVER` parameter from the Extract parameter file and restart Extract. Remember to set the parameter after the standby becomes available.

> **Note:**
>
> If extracting from a primary database continues while the standby is unavailable, a data loss failover could result after the standby becomes available, and not all the primary redo was applied before a failover. The GoldenGate target database will contain data that does not exist in the source database.

If the Extract process has been assigned an auto restart profile, as documented in Cloud: Oracle GoldenGate Microservices Architecture on Oracle Exadata Database Service Configuration Best Practices , after a Data Guard role transition, the Extract process automatically restarts. Extract continues to mine redo data from the new primary database, ignoring the current state of the new standby database, until a default 5 minute timeout period expires. After this time, if the standby is not available Extract will abend with the following errors:

```
INFO OGG-25053 Timeout waiting for 300 seconds for standby database
reinstatement. Now enforcing HANDLEDLFAILOVER.

ERROR OGG-06219 Unable to extract data from the Logmining server OGG$CAP_XXXXX.

ERROR OGG-02078 Extract encountered a fatal error in a processing thread and is
abending.
```

Extract continues attempting to automatically restart, based on the Oracle GoldenGate Microservices auto restart profile, and fails because it reaches the `HANDLEDLFAILOVER` timeout, until the number of retries is reached or the new standby database becomes available.

During the timeout period following a database role transition, the `HANDLEDLFAILOVER` parameter is automatically suspended, so data is replicated to the Oracle GoldenGate replica database without consideration of the source standby database not being kept up to date. The timeout period for the standby database to start up before Extract abends can be adjusted using the Extract parameter `TRANLOGOPTIONS DLFAILOVER_TIMEOUT`.

It is recommended that you leave `DLFAILOVER_TIMEOUT` at the default of 5 minutes to allow the old primary to convert to a standby. If the new standby database will be unavailable for an extended period of time or completely gone, then to ensure that Extract starts and remains running, you must remove the `HANDLEDLFAILOVER` parameter from the Extract parameter file. After removing the parameter, Extract no longer waits until redo has been applied to the standby database before extracting the data.

During the time it takes for the standby database to come back online and apply all of the redo from the primary database, there is data divergence between it and the Oracle GoldenGate replica database. This divergence is resolved when the standby database is up to date. At this point you can add the `HANDLEDLFAILOVER` parameter back into the integrated Extract process parameter file, and then stop and restart the Extract.

When Oracle Data Guard Fast Start Failover is disabled, such that the broker can automatically fail over to a standby database in the event of loss of the primary database, you must specify an additional integrated Extract parameter:

```
TRANLOGOPTIONS FAILOVERTARGETDESTID n
```

This parameter identifies which standby database the Oracle GoldenGate Extract process must remain behind, with regards to not extracting redo data that has not yet been applied to the standby database.

If Oracle Data Guard Fast Start Failover is disabled, and you don't specify the additional integrated Extract parameter `FAILOVERTARGETDESTID`, the extract will abend with the following errors:

```
ERROR OGG-06219 Unable to extract data from the Logmining server OGG$CAP_XXXXX.

ERROR OGG-02078 Extract encountered a fatal error in a processing thread and is
abending.
```

Troubleshooting ACFS Replication

The health of ACFS replication is determined by the `acfsutil repl util verifyprimary/verifystandby` commands. These commands are called by the example CRS action scripts *acfs_primary.scr* and *acfs_standby.scr*, but they are also implicitly called during a file system role transition.

Both commands will return a value of '0' if there are no problems detected. If a non-zero value is returned, run the same command with verbose flag to see comprehensive output of the verification tests.

As the grid user on the standby GGHub system, verify the ACFS replication with the primary GGHub:

```
[grid@gghub_stby1]$ acfsutil repl util verifyprimary /mnt/acfs_gg -v

- Attempting to ping clust1-vip1
- ping successful
- Attempting to ssh
  '/usr/bin/ssh -o BatchMode=true -o Ciphers=aes128-ctr -o ConnectTimeout=3
-x oracle@clust1-vip1 true 2>&1'
- ssh output: Host key verification failed.
- ssh output: Host key verification failed.
- ssh attempt failed, ret=255
verifyprimary return code: 255
```

The errors reported by the verify command, `Host key verification failed`, clearly showing why it failed. In this example, there is a problem with the ssh configuration between the standby and the primary file system GGHubs. Once the problem has been resolved, rerun the verify commands to ensure there are no further problems.

After a failover has completed, it is recommended to check the acfsutil trace files for the reason behind the failover. The acfsutil trace files are located in the CRS trace file directory, which defaults to `/u01/app/grid/diag/crs/`hostname`/crs/trace/crsd_scriptagent_grid.trc`.

Below are some common failures that can occur with incorrect ACFS replication configuration.

SSH daemon is shutdown or not configured to run on the VIP

When using an application VIP on the ACFS primary and standby GGHubs, the ssh daemon must be configured to listen for incoming connections on the VIP address. If this configuration is not done, or the ssh daemon is not running on either of the current primary/standby hosts the verifyprimary or verifystandby commands will fail with the following error.

As the grid user on the primary GGHub system, verify the ACFS recplication with the standby GGHub:

```
[grid@gghub_prim1]$ acfsutil repl util verifystandby /mnt/acfs_gg -v


- Attempting to ping gghubstby.goldengate.com
- ping successful
- Attempting to ssh
  '/usr/bin/ssh -o BatchMode=true   -o Ciphers=aes128-ctr -o
ConnectTimeout=3   -x oracle@gghub_stby-avip true 2>&1'

- ssh output: ssh: connect to host gghub_stby1 port 22: Connection refused
- ssh output: ssh: connect to host gghub_stby2 port 22: Connection refused

- ssh attempt failed, ret=255
verifystandby return code: 255
```

As the grid user on the standby GGHub system, check that the resource application VIP and sshd_restart are running and restart them if not:

```
[grid@gghub_stby1 ~]$ crsctl stat res -w "TYPE co app.appviptypex2"

NAME=gghubstby
TYPE=app.appviptypex2.type
TARGET=OFFLINE
STATE=OFFLINE

[grid@gghub_stby1 ~]$ crsctl start res gghubstby

CRS-2672: Attempting to start 'gghubstby' on 'gghub_stby1'
CRS-2676: Start of 'gghubstby' on 'gghub_stby1' succeeded
CRS-2672: Attempting to start 'sshd_restart' on 'gghub_stby1'
CRS-2676: Start of 'sshd_restart' on 'gghub_stby1' succeeded
```

Check that `acfsutil repl verifystandby/verifyprimary` returns a result of ‘0’ from both the primary and standby host.

Primary ACFS background resources are not running

1. The primary or standby ACFS servers are not accessible
2. ACFS Replication ssh user problem
3. SSH Host key verification failed

Troubleshooting Oracle GoldenGate

There may be occasions when GoldenGate processes are not successfully started on an Oracle RAC node. There are number of files generated by GoldenGate, XAG, and CRS that should be reviewed to determine the cause of the problem.

Below is a list of important log and trace files, along with their example locations and some example output.

XAG log file

Location: <XAG installation directory>/log/<hostname>

Example location: /u01/app/grid/xag/log/`hostname`

File name: agctl_goldengate_grid.trc

Contains all commands executed with agctl along with the output from the commands, including those that CRS executes.

```
2022-04-18 11:52:21: stop resource success
2022-04-18 11:52:38: agctl start goldengate <instance_name>
2022-04-18 11:52:38: executing cmd: /u01/app/19.0.0.0/grid/bin/crsctl status
res xag.<INSTANCE_NAME>.goldengate
2022-04-18 11:52:38: executing cmd: /u01/app/19.0.0.0/grid/bin/crsctl status
res xag.<INSTANCE_NAME>.goldengate -f
2022-04-18 11:52:38: executing cmd: /u01/app/19.0.0.0/grid/bin/crsctl start
resource xag.<INSTANCE_NAME>.goldengate -f
2022-04-18 11:52:45: Command output:
>  CRS-2672: Attempting to start 'xag.<INSTANCE_NAME>.goldengate' on 'exadb-
node1'
>  CRS-2676: Start of 'xag.<INSTANCE_NAME>.goldengate' on 'exadb-node1'
succeeded
>End Command output
2022-04-18 11:52:45: start resource success
```

XAG GoldenGate instance trace file

Location: <XAG installation directory>/log/<hostname>

Example location: /u01/app/grid/xag/log/`hostname`

File name: <GoldenGate_instance_name>_agent_goldengate.trc

It contains the output from the commands executed by agctl, the environment variables used, and any debug output enabled for the underlying commands.

```
2022-04-18 12:14:46: Exported ORACLE_SID ggdg1
2022-04-18 12:14:46: Exported GGS_HOME /u01/oracle/goldengate/gg21c_MS
2022-04-18 12:14:46: Exported OGG_CONF_HOME /mnt/dbfs/goldengate/deployments/
ggsm01/etc/conf
2022-04-18 12:14:46: Exported LD_LIBRARY_PATH
/u01/oracle/goldengate/gg21c_MS:/u01/app/19.0.0.0/grid/lib:/etc/
ORCLcluster/lib
2022-04-18 12:14:46: Exported LD_LIBRARY_PATH_64 /u01/oracle/goldengate/
gg21c_MS
2022-04-18 12:14:46: Exported LIBPATH /u01/oracle/goldengate/gg21c_MS
2022-04-18 12:14:46: ogg input = {"oggHome":"/u01/oracle/goldengate/
gg21c_MS","serviceManager":{"oggConfHome":"/mnt/dbfs/goldengate/deployments/
ggsm01/etc/
conf","portNumber":9100},"username":"<username>","credential":"*****"}
```

```
2022-04-18 12:14:46: About to exec /u01/oracle/goldengate/gg21c_MS/bin/
XAGTask HealthCheck
2022-04-18 12:14:47: XAGTask retcode = 0
```

CRS trace file

Location: /u01/app/grid/diag/crs/<hostname>/crs/trace

Example location: /u01/app/grid/diag/crs/`hostname`/crs/trace

File name: crsd_scriptagent_oracle.trc

Contains the output created by any CRS resource action scripts, like XAG or dbfs_mount. This trace file is crucial to determining why DBFS or GoldenGate did not start on a RAC node.

```
2022-04-18 11:52:38.634 :    AGFW:549631744: {1:30281:59063} Agent received
the message: RESOURCE_START[xag.<INSTANCE_NAME>.goldengate 1 1] ID
4098:4125749
2022-04-18 11:52:38.634 :    AGFW:549631744: {1:30281:59063} Preparing START
command for: xag.<INSTANCE_NAME>.goldengate 1 1
2022-04-18 11:52:38.634 :    AGFW:549631744: {1:30281:59063}
xag.<INSTANCE_NAME>.goldengate 1 1 state changed from: OFFLINE to: STARTING
2022-04-18 11:52:38.634 :CLSDYNAM:558036736: [xag.<INSTANCE_NAME>.goldengate]
{1:30281:59063} [start] Executing action script: /u01/oracle/XAG_MA/bin/
aggoldengatescaas[start]
2022-04-18 11:52:38.786 :CLSDYNAM:558036736: [xag.<INSTANCE_NAME>.goldengate]
{1:30281:59063} [start] GG agent running command 'start' on
xag.<INSTANCE_NAME>.goldengate
2022-04-18 11:52:42.140 :CLSDYNAM:558036736: [xag.<INSTANCE_NAME>.goldengate]
{1:30281:59063} [start] ServiceManager fork pid = 265747
2022-04-18 11:52:42.140 :CLSDYNAM:558036736: [xag.<INSTANCE_NAME>.goldengate]
{1:30281:59063} [start] Waiting for /mnt/dbfs/goldengate/deployments/
ggsm01/var/run/ServiceManager.pid
2022-04-18 11:52:42.140 :CLSDYNAM:558036736: [xag.<INSTANCE_NAME>.goldengate]
{1:30281:59063} [start] Waiting for SM to start
2022-04-18 11:52:42.140 :CLSDYNAM:558036736: [xag.<INSTANCE_NAME>.goldengate]
{1:30281:59063} [start] ServiceManager PID = 265749
2022-04-18 11:52:43.643 :CLSDYNAM:558036736: [xag.<INSTANCE_NAME>.goldengate]
{1:30281:59063} [start] XAGTask retcode = 0
2022-04-18 11:52:43.643 :CLSDYNAM:558036736: [xag.<INSTANCE_NAME>.goldengate]
{1:30281:59063} [start] XAG HealthCheck after start returned 0
2022-04-18 11:52:43.643 :    AGFW:558036736: {1:30281:59063} Command: start
for resource: xag.<INSTANCE_NAME>.goldengate 1 1 completed with status:
SUCCESS
2022-04-18 11:52:43.643 :CLSDYNAM:558036736: [xag.<INSTANCE_NAME>.goldengate]
{1:30281:59063} [check] Executing action script: /u01/oracle/XAG_MA/bin/
aggoldengatescaas[check]
2022-04-18 11:52:43.644 :    AGFW:549631744: {1:30281:59063} Agent sending
reply for: RESOURCE_START[xag.<INSTANCE_NAME>.goldengate 1 1] ID 4098:4125749
2022-04-18 11:52:43.795 :CLSDYNAM:558036736: [xag.<INSTANCE_NAME>.goldengate]
{1:30281:59063} [check] GG agent running command 'check' on
xag.<INSTANCE_NAME>.goldengate
2022-04-18 11:52:45.548 :CLSDYNAM:558036736: [xag.<INSTANCE_NAME>.goldengate]
{1:30281:59063} [check] XAGTask retcode = 0
2022-04-18 11:52:45.548 :    AGFW:549631744: {1:30281:59063}
xag.<INSTANCE_NAME>.goldengate 1 1 state changed from: STARTING to: ONLINE
```

GoldenGate deployment log files

Location: <Goldengate_deployment_directory>/<instance_name>/var/log

Example location: /mnt/dbfs/goldengate/deployments/<instance_name>/var/log

File names: adminsrvr.log, recvsrvr.log, pmsrvr.log, distsrvr.log

Contains the output of start, stop, and status checks of the Oracle GoldenGate deployment processes (Administration Server, Distribution Server, Receiver Server, and Performance Metrics Server).

```
2022-04-18T11:52:42.645-0400 INFO | Setting deploymentName to
'<instance_name>'. (main)
2022-04-18T11:52:42.665-0400 INFO | Read SharedContext from store for length
19 of file '/mnt/dbfs/goldengate/deployments/<instance_name>/var/lib/conf/
adminsrvr-resources.dat'. (main)
2022-04-18T11:52:42.723-0400 INFO | XAG Integration enabled (main)
2022-04-18T11:52:42.723-0400 INFO | Configuring security. (main)
2022-04-18T11:52:42.723-0400 INFO | Configuring user authorization secure
store path as '/mnt/dbfs/goldengate/deployments/<instance_name>/var/lib/
credential/secureStore/'. (main)
2022-04-18T11:52:42.731-0400 INFO | Configuring user authorization as
ENABLED. (main)
2022-04-18T11:52:42.749-0400 INFO | Set network configuration. (main)
2022-04-18T11:52:42.749-0400 INFO | Asynchronous operations are enabled with
default synchronous wait time of 30 seconds (main)
2022-04-18T11:52:42.749-0400 INFO | HttpServer configuration complete. (main)
2022-04-18T11:52:42.805-0400 INFO | SIGHUP handler installed. (main)
2022-04-18T11:52:42.813-0400 INFO | SIGINT handler installed. (main)
2022-04-18T11:52:42.815-0400 INFO | SIGTERM handler installed. (main)
2022-04-18T11:52:42.817-0400 WARN | Security is configured as 'disabled'.
(main)
2022-04-18T11:52:42.818-0400 INFO | Starting service listener... (main)
2022-04-18T11:52:42.819-0400 INFO | Mapped 'ALL' interface to address
'ANY:9101' with default IPV4/IPV6 options identified by 'exadb-node1.domain'.
(main)
2022-04-18T11:52:42.821-0400 INFO | Captured 1 interface host names: 'exadb-
node1.domain' (main)
2022-04-18T11:52:42.824-0400 INFO | The Network ipACL specification is empty.
Accepting ANY address on ALL interfaces. (main)
2022-04-18T11:52:42.826-0400 INFO | Server started at
2022-04-18T11:52:42.827-05:00 (2022-04-18T15:52:42.827Z GMT) (main)
```

GoldenGate report files

Location: <Goldengate_deployment_directory>/<instance_name>/var/lib/report

Example location: /mnt/dbfs/goldengate/deployments/<instance_name>/var/lib/report

The GoldenGate report files contain important information, warning messages, and errors for all GoldenGate processes, including the Manager processes. If any of the GoldenGate processes fail to start or abend when running, the process report file will contain important information that can be used to determine the cause of the failure.

```
2022-04-23 13:01:50  ERROR   OGG-00446  Unable to lock file " /mnt/acfs_gg/
deployments/<instance_name>/var/lib/checkpt/EXT_1A.cpe" (error 95, Operation
```

```
not supported).
2022-04-23 13:01:50  ERROR   OGG-01668  PROCESS ABENDING.
```

Troubleshooting Oracle GoldenGate on Oracle RAC

There may be occasions when Oracle GoldenGate processes are not successfully started on an Oracle RAC node. Several files generated by Oracle GoldenGate, XAG, and CRS should be reviewed to determine the cause of the problem.

Below is a list of important log and trace files, their example locations, and some examples of output.

XAG log file

Location: <XAG installation directory>/log/<hostname>

Example location: /u01/app/grid/xag/log/`hostname`

File name: agctl_goldengate_grid.trc

Contains all commands executed with agctl along with the output from the commands, including those that CRS executes.

Example:

```
2022-04-18 11:52:21: stop resource success
2022-04-18 11:52:38: agctl start goldengate <instance_name>
2022-04-18 11:52:38: executing cmd: /u01/app/19.0.0.0/grid/bin/crsctl status
res xag.<INSTANCE_NAME>.goldengate
2022-04-18 11:52:38: executing cmd: /u01/app/19.0.0.0/grid/bin/crsctl status
res xag.<INSTANCE_NAME>.goldengate -f
2022-04-18 11:52:38: executing cmd: /u01/app/19.0.0.0/grid/bin/crsctl start
resource xag.<INSTANCE_NAME>.goldengate -f
2022-04-18 11:52:45: Command output:
>  CRS-2672: Attempting to start 'xag.<INSTANCE_NAME>.goldengate' on 'exadb-
node1'
>  CRS-2676: Start of 'xag.<INSTANCE_NAME>.goldengate' on 'exadb-node1'
succeeded
>End Command output
2022-04-18 11:52:45: start resource success
```

XAG GoldenGate instance trace file

Location: <XAG installation directory>/log/<hostname>

Example location: /u01/app/grid/xag/log/`hostname`

File name: <GoldenGate_instance_name>_agent_goldengate.trc

It contains the output from the commands executed by agctl, the environment variables used, and any debug output enabled for the underlying commands.

Example:

```
2022-04-18 12:14:46: Exported ORACLE_SID ggdg1
2022-04-18 12:14:46: Exported GGS_HOME /u01/oracle/goldengate/gg21c_MS
2022-04-18 12:14:46: Exported OGG_CONF_HOME /mnt/dbfs/goldengate/deployments/
ggsm01/etc/conf
```

```
2022-04-18 12:14:46: Exported LD_LIBRARY_PATH
/u01/oracle/goldengate/gg21c_MS:/u01/app/19.0.0.0/grid/lib:/etc/
ORCLcluster/lib
2022-04-18 12:14:46: Exported LD_LIBRARY_PATH_64 /u01/oracle/goldengate/
gg21c_MS
2022-04-18 12:14:46: Exported LIBPATH /u01/oracle/goldengate/gg21c_MS
2022-04-18 12:14:46: ogg input = {"oggHome":"/u01/oracle/goldengate/
gg21c_MS","serviceManager":{"oggConfHome":"/mnt/dbfs/goldengate/deployments/
ggsm01/etc/conf","portNumber":9100},"username":"admin","credential":"xyz"}
2022-04-18 12:14:46: About to exec /u01/oracle/goldengate/gg21c_MS/bin/
XAGTask HealthCheck
2022-04-18 12:14:47: XAGTask retcode = 0
```

CRS trace file

Location: /u01/app/grid/diag/crs/<hostname>/crs/trace

Example location: /u01/app/grid/diag/crs/`hostname`/crs/trace

File name: crsd_scriptagent_oracle.trc

Contains the output created by any CRS resource action scripts, like XAG or dbfs_mount. This trace file is crucial to determining why DBFS or GoldenGate did not start on a RAC node.

Example:

```
2022-04-18 11:52:38.634 :    AGFW:549631744: {1:30281:59063} Agent received
the message: RESOURCE_START[xag.<INSTANCE_NAME>.goldengate 1 1] ID
4098:4125749
2022-04-18 11:52:38.634 :    AGFW:549631744: {1:30281:59063} Preparing START
command for: xag.<INSTANCE_NAME>.goldengate 1 1
2022-04-18 11:52:38.634 :    AGFW:549631744: {1:30281:59063}
xag.<INSTANCE_NAME>.goldengate 1 1 state changed from: OFFLINE to: STARTING
2022-04-18 11:52:38.634 :CLSDYNAM:558036736: [xag.<INSTANCE_NAME>.goldengate]
{1:30281:59063} [start] Executing action script: /u01/oracle/XAG_MA/bin/
aggoldengatescaas[start]
2022-04-18 11:52:38.786 :CLSDYNAM:558036736: [xag.<INSTANCE_NAME>.goldengate]
{1:30281:59063} [start] GG agent running command 'start' on
xag.<INSTANCE_NAME>.goldengate
2022-04-18 11:52:42.140 :CLSDYNAM:558036736: [xag.<INSTANCE_NAME>.goldengate]
{1:30281:59063} [start] ServiceManager fork pid = 265747
2022-04-18 11:52:42.140 :CLSDYNAM:558036736: [xag.<INSTANCE_NAME>.goldengate]
{1:30281:59063} [start] Waiting for /mnt/dbfs/goldengate/deployments/
ggsm01/var/run/ServiceManager.pid
2022-04-18 11:52:42.140 :CLSDYNAM:558036736: [xag.<INSTANCE_NAME>.goldengate]
{1:30281:59063} [start] Waiting for SM to start
2022-04-18 11:52:42.140 :CLSDYNAM:558036736: [xag.<INSTANCE_NAME>.goldengate]
{1:30281:59063} [start] ServiceManager PID = 265749
2022-04-18 11:52:43.643 :CLSDYNAM:558036736: [xag.<INSTANCE_NAME>.goldengate]
{1:30281:59063} [start] XAGTask retcode = 0
2022-04-18 11:52:43.643 :CLSDYNAM:558036736: [xag.<INSTANCE_NAME>.goldengate]
{1:30281:59063} [start] XAG HealthCheck after start returned 0
2022-04-18 11:52:43.643 :    AGFW:558036736: {1:30281:59063} Command: start
for resource: xag.<INSTANCE_NAME>.goldengate 1 1 completed with status:
SUCCESS
2022-04-18 11:52:43.643 :CLSDYNAM:558036736: [xag.<INSTANCE_NAME>.goldengate]
{1:30281:59063} [check] Executing action script: /u01/oracle/XAG_MA/bin/
```

```
aggoldengatescaas[check]
2022-04-18 11:52:43.644 :    AGFW:549631744: {1:30281:59063} Agent sending
reply for: RESOURCE_START[xag.<INSTANCE_NAME>.goldengate 1 1] ID 4098:4125749
2022-04-18 11:52:43.795 :CLSDYNAM:558036736: [xag.<INSTANCE_NAME>.goldengate]
{1:30281:59063} [check] GG agent running command 'check' on
xag.<INSTANCE_NAME>.goldengate
2022-04-18 11:52:45.548 :CLSDYNAM:558036736: [xag.<INSTANCE_NAME>.goldengate]
{1:30281:59063} [check] XAGTask retcode = 0
2022-04-18 11:52:45.548 :    AGFW:549631744: {1:30281:59063}
xag.<INSTANCE_NAME>.goldengate 1 1 state changed from: STARTING to: ONLINE
```

GoldenGate deployment log files

Location: <Goldengate_deployment_directory>/<instance_name>/var/log

Example location: /mnt/dbfs/goldengate/deployments/<instance_name>/var/log

File names: adminsrvr.log, recvsrvr.log, pmsrvr.log, distsrvr.log

Contains the output of start, stop, and status checks of the Oracle GoldenGate deployment processes (Administration Server, Distribution Server, Receiver Server, and Performance Metrics Server).

Example:

```
2022-04-18T11:52:42.645-0400 INFO | Setting deploymentName to
'<instance_name>'. (main)
2022-04-18T11:52:42.665-0400 INFO | Read SharedContext from store for length
19 of file '/mnt/dbfs/goldengate/deployments/<instance_name>/var/lib/conf/
adminsrvr-resources.dat'. (main)
2022-04-18T11:52:42.723-0400 INFO | XAG Integration enabled (main)
2022-04-18T11:52:42.723-0400 INFO | Configuring security. (main)
2022-04-18T11:52:42.723-0400 INFO | Configuring user authorization secure
store path as '/mnt/dbfs/goldengate/deployments/<instance_name>/var/lib/
credential/secureStore/'. (main)
2022-04-18T11:52:42.731-0400 INFO | Configuring user authorization as
ENABLED. (main)
2022-04-18T11:52:42.749-0400 INFO | Set network configuration. (main)
2022-04-18T11:52:42.749-0400 INFO | Asynchronous operations are enabled with
default synchronous wait time of 30 seconds (main)
2022-04-18T11:52:42.749-0400 INFO | HttpServer configuration complete. (main)
2022-04-18T11:52:42.805-0400 INFO | SIGHUP handler installed. (main)
2022-04-18T11:52:42.813-0400 INFO | SIGINT handler installed. (main)
2022-04-18T11:52:42.815-0400 INFO | SIGTERM handler installed. (main)
2022-04-18T11:52:42.817-0400 WARN | Security is configured as 'disabled'.
(main)
2022-04-18T11:52:42.818-0400 INFO | Starting service listener... (main)
2022-04-18T11:52:42.819-0400 INFO | Mapped 'ALL' interface to address
'ANY:9101' with default IPV4/IPV6 options identified by 'exadb-node1.domain'.
(main)
2022-04-18T11:52:42.821-0400 INFO | Captured 1 interface host names: 'exadb-
node1.domain' (main)
2022-04-18T11:52:42.824-0400 INFO | The Network ipACL specification is empty.
Accepting ANY address on ALL interfaces. (main)
2022-04-18T11:52:42.826-0400 INFO | Server started at
2022-04-18T11:52:42.827-05:00 (2022-04-18T15:52:42.827Z GMT) (main)
```

GoldenGate report files

Location: <Goldengate_deployment_directory>/<instance_name>/var/lib/report

Example location: /mnt/dbfs/goldengate/deployments/<instance_name>/var/lib/report

The GoldenGate report files contain important information, warning messages, and errors for all GoldenGate processes, including the Manager processes. If any of the GoldenGate processes fail to start or abend when running, the process report file will contain important information that can be used to determine the cause of the failure.

Example errors from an Extract report file:

```
2022-04-23 13:01:50  ERROR   OGG-00446  Unable to lock file " /mnt/acfs_gg/
deployments/<instance_name>/var/lib/checkpt/EXT_1A.cpe" (error 95, Operation
not supported).
2022-04-23 13:01:50  ERROR   OGG-01668  PROCESS ABENDING.
```

Example Configuration Problems

Below are some configuration problems that can be encountered with GoldenGate in a RAC environment and how to diagnose and resolve them.

Incorrect parameter settings in the mount-dbfs.conf file

When XAG fails to mount DBFS, the failure will be reported either on the command line (if you are running the manual agctl command) or in the XAG log file:

```
$ agctl start goldengate <instance_name> --node exadb-node1

CRS-2672: Attempting to start 'dbfs_mount' on 'exadb-node1'
CRS-2674: Start of 'dbfs_mount' on 'exadb-node1' failed
CRS-2679: Attempting to clean 'dbfs_mount' on 'exadb-node1'
CRS-2681: Clean of 'dbfs_mount' on 'exadb-node1' succeeded
CRS-4000: Command Start failed, or completed with errors.
```

The XAG log file (agctl_goldengate_grid.trc) has the advantage that it shows timestamps that can be used when looking at other log or trace files:

```
2022-04-19 15:32:16: executing cmd: /u01/app/19.0.0.0/grid/bin/crsctl start
resource xag.<INSTANCE_NAME>.goldengate -f -n exadb-node1
2022-04-19 15:32:19: Command output:
>  CRS-2672: Attempting to start 'dbfs_mount' on 'exadb-node1'
>  CRS-2674: Start of 'dbfs_mount' on 'exadb-node1' failed
>  CRS-2679: Attempting to clean 'dbfs_mount' on 'exadb-node1'
>  CRS-2681: Clean of 'dbfs_mount' on 'exadb-node1' succeeded
>  CRS-4000: Command Start failed, or completed with errors.
>End Command output
2022-04-19 15:32:19: start resource failed rc=1
```

Next, check the CRS trace file (crsd_scriptagent_oracle.trc), which shows why DBFS failed to mount. Below are some example errors caused by incorrect parameter settings in the mount-dbfs.conf file.

- Incorrect DBNAME

```
2022-04-19 15:32:16.679 :    AGFW:1190405888: {1:30281:17383} dbfs_mount
 1 1 state changed from: UNKNOWN to: STARTING
2022-04-19 15:32:16.680 :CLSDYNAM:1192507136: [dbfs_mount]{1:30281:17383}
[start]
 Executing action script: /u01/oracle/scripts/mount-dbfs.sh[start]
2022-04-19 15:32:16.732 :CLSDYNAM:1192507136: [dbfs_mount]{1:30281:17383}
[start]
 mount-dbfs.sh mounting DBFS at /mnt/dbfs from database ggdg
2022-04-19 15:32:17.883 :CLSDYNAM:1192507136: [dbfs_mount]{1:30281:17383}
[start]
 ORACLE_SID is
2022-04-19 15:32:17.883 :CLSDYNAM:1192507136: [dbfs_mount]{1:30281:17383}
[start]
 No running ORACLE_SID available on this host, exiting
2022-04-19 15:32:17.883 :    AGFW:1192507136: {1:30281:17383} Command:
start for
 resource: dbfs_mount 1 1 completed with invalid status: 2
```

- Incorrect MOUNT_POINT

```
2022-04-19 16:45:14.534 :    AGFW:1734321920: {1:30281:17604} dbfs_mount
 1 1 state changed from: UNKNOWN to: STARTING
2022-04-19 16:45:14.535 :CLSDYNAM:1736423168: [dbfs_mount]{1:30281:17604}
[start]
 Executing action script: /u01/oracle/scripts/mount-dbfs.sh[start]
2022-04-19 16:45:14.586 :CLSDYNAM:1736423168: [dbfs_mount]{1:30281:17604}
[start]
 mount-dbfs.sh mounting DBFS at /mnt/dbfs from database ggdgs
2022-04-19 16:45:15.638 :CLSDYNAM:1736423168: [dbfs_mount]{1:30281:17604}
[start]
 ORACLE_SID is ggdg1
2022-04-19 16:45:15.738 :CLSDYNAM:1736423168: [dbfs_mount]{1:30281:17604}
[start]
 spawning dbfs_client command using SID ggdg1
2022-04-19 16:45:20.745 :CLSDYNAM:1736423168: [dbfs_mount]{1:30281:17604}
[start]
 fuse: bad mount point `/mnt/dbfs': No such file or directory
2022-04-19 16:45:21.747 :CLSDYNAM:1736423168: [dbfs_mount]{1:30281:17604}
[start]
 Start - OFFLINE
2022-04-19 16:45:21.747 :    AGFW:1736423168: {1:30281:17604} Command:
start for
 resource: dbfs_mount 1 1 completed with status: FAIL
```

- Incorrect DBFS_USER or DBFS_PASSWD

```
2022-04-19 16:47:47.855 :    AGFW:1384478464: {1:30281:17671} dbfs_mount
 1 1 state changed from: UNKNOWN to: STARTING
2022-04-19 16:47:47.856 :CLSDYNAM:1386579712: [dbfs_mount]{1:30281:17671}
[start]
 Executing action script: /u01/oracle/scripts/mount-dbfs.sh[start]
2022-04-19 16:47:47.908 :CLSDYNAM:1386579712: [dbfs_mount]{1:30281:17671}
[start]
 mount-dbfs.sh mounting DBFS at /mnt/dbfs from database ggdgs
```

```
2022-04-19 16:47:48.959 :CLSDYNAM:1386579712: [dbfs_mount]{1:30281:17671}
[start]
 ORACLE_SID is ggdg1
2022-04-19 16:47:49.010 :CLSDYNAM:1386579712: [dbfs_mount]{1:30281:17671}
[start]
 spawning dbfs_client command using SID ggdg1
2022-04-19 16:47:55.118 :CLSDYNAM:1386579712: [dbfs_mount]{1:30281:17671}
[start]
 Fail to connect to database server. Error: ORA-01017: invalid username/
password;
 logon denied
2022-04-19 16:47:55.118 :CLSDYNAM:1386579712: [dbfs_mount]{1:30281:17671}
[start]
2022-04-19 16:47:56.219 :CLSDYNAM:1386579712: [dbfs_mount]{1:30281:17671}
[start]
 Start - OFFLINE
2022-04-19 16:47:56.220 :    AGFW:1386579712: {1:30281:17671} Command:
start for
 resource: dbfs_mount 1 1 completed with status: FAIL
```

- Incorrect ORACLE_HOME

```
2022-04-19 16:50:38.952 :    AGFW:567502592: {1:30281:17739} dbfs_mount
 1 1 state changed from: UNKNOWN to: STARTING
2022-04-19 16:50:38.953 :CLSDYNAM:569603840: [dbfs_mount]{1:30281:17739}
[start]
 Executing action script: /u01/oracle/scripts/mount-dbfs.sh[start]
2022-04-19 16:50:39.004 :CLSDYNAM:569603840: [dbfs_mount]{1:30281:17739}
[start]
 mount-dbfs.sh mounting DBFS at /mnt/dbfs from database ggdgs
2022-04-19 16:50:39.004 :CLSDYNAM:569603840: [dbfs_mount]{1:30281:17739}
[start]
 /u01/oracle/scripts/mount-dbfs.sh: line 136:
 /u01/app/oracle/product/19.0.0.0/rdbms/bin/srvctl: No such file or
directory
2022-04-19 16:50:39.004 :CLSDYNAM:569603840: [dbfs_mount]{1:30281:17739}
[start]
 /u01/oracle/scripts/mount-dbfs.sh: line 139:
 /u01/app/oracle/product/19.0.0.0/rdbms/bin/srvctl: No such file or
directory
2022-04-19 16:50:39.004 :CLSDYNAM:569603840: [dbfs_mount]{1:30281:17739}
[start]
 ORACLE_SID is
2022-04-19 16:50:39.004 :CLSDYNAM:569603840: [dbfs_mount]{1:30281:17739}
[start]
 No running ORACLE_SID available on this host, exiting
2022-04-19 16:50:39.004 :    AGFW:569603840: {1:30281:17739} Command:
start for
 resource: dbfs_mount 1 1 completed with invalid status: 2
```

To resolve these configuration issues, set the correct parameter values in mount-dbfs.conf.

Problems with file locking on DBFS

If using Oracle Database 12c Release 2 (12.2) and the nolock DBFS mount option is not used, there can be problems with GoldenGate processes trying to lock checkpoint or trail files. The

same problem will be encountered if using Oracle Database 11g Release 2 (11.2.0.4) or 12c Release 1 (12.1) with a patch for bug 22646150 applied. This patch changes how DBFS handles file locking to match Oracle Database 12c Release 2 (12.2). To add the nolock DBFS mount option, a patch for bug 27056711 must be applied to the database. If the patch for bug 22646150 has not been applied to the database, the patch for bug 27056711 and the nolock mount option is not required.

Below is an example of diagnosing a GoldenGate Microservices Architecture locking problem.

When starting a deployment with XAG, one or more processes may not start due to detecting a locking conflict on one or more files. This will often occur after a RAC node failover where the deployment did not get a chance to shut down cleanly.

When one of the deployment server processes fails to start (Administration Server, Performance Metrics Server, Distribution Server, Receiver Server, or Service Manager), check the log file for the particular server located in the deployment var/log directory.

For example, the log file /mnt/dbfs/goldengate/deployments/<INSTANCE_NAME>/var/log/pmsrvr.log shows the following error on startup:

```
2022-04-11T12:41:57.619-0700 ERROR| SecureStore failed on open after
 retrying due to extended file lock. (main)
2022-04-11T12:41:57.619-0700 ERROR| SecureStore failed to close (28771).
(main)
2022-04-11T12:41:57.619-0700 INFO | Set network configuration. (main)
2022-04-11T12:41:57.619-0700 INFO | Asynchronous operations are enabled with
default
 synchronous wait time of 30 seconds (main)
2022-04-11T12:41:57.619-0700 INFO | HttpServer configuration complete. (main)
2022-04-11T12:42:07.674-0700 ERROR| Unable to lock process file, Error is
[1454]
 - OGG-01454 (main)
2022-04-11T12:42:07.675-0700 ERROR| Another Instance of PM Server is Already
Running
 (main)
```

An Extract process will report start-up failures in the ER-events.log logfile located in the deployment log file directory.

For example, /mnt/dbfs/goldengate/deployments/<instance_name>/var/log/ER-events.log shows the following error:

```
2022-04-11T00:14:56.845-0700  ERROR   OGG-01454  Oracle GoldenGate Capture for
 Oracle, EXT1.prm:  Unable to lock file
 "/mnt/dbfs/goldengate/deployments/<instance_name>/var/run/EXT1.pce" (error
11, Resource
 temporarily unavailable).  Lock currently held by process id (PID) 237495.
2022-04-11T00:14:56.861-0700  ERROR   OGG-01668  Oracle GoldenGate Capture
for Oracle,
 EXT1.prm:  PROCESS ABENDING.
```

Next, check to ensure the process failing to start up is not running on any of the RAC nodes.

Example:

```
$ ps -ef|grep EXT1|grep -v grep
```

Once it has been determined that the process is not running, the deployment must be shutdown cleanly, the file system unmounted, and the correct DBFS patch applied.

Example:

```
$ agctl stop goldengate <INSTANCE_NAME>
$ crsctl stop resource dbfs_mount
```

Check the DBFS mount options:

```
$ ps -ef|grep dbfs_client

oracle   204017      1  0 14:37 ?
       00:00:00 /u01/app/oracle/product/19.1.0.0/dbhome_1/bin/dbfs_client
dbfs@dbfs.local
 -o allow_other,failover,direct_io /mnt/dbfs
```

It is clear the nolock mount option was not used, which leads to the locking errors.

Use the guidelines above to determine if a DBFS patch is required. After which, add the nolock mount option to the mount-dbfs.conf file on all Oracle RAC nodes that are part of the deployment.

Example:

```
MOUNT_OPTIONS=allow_other,direct_io,failover,nolock
```

Finally, restart the deployment:

```
$ agctl start goldengate <INSTANCE_NAME>
```

Part VI
Oracle Database Cloud Best Practices

28

Overview of Oracle Database Cloud Best Practices

Oracle Cloud and MAA collaborate closely to enable customer success using the varies Oracle Database Cloud services.

This partnership ensures

- Databases, Grid Infrastructure, Exadata, and infrastructure are deployed and configured with MAA best practices in the Oracle cloud.
- Cloud life cycle practices such as patching, backup and restore, disaster recovery, and pluggable database management incorporate MAA optimizations as new features and capabilities are introduced into the Oracle cloud.
- Oracle owns and manages network, system, and Exadata infrastructure using MAA practices and optimizations.
- Oracle high availability and disaster recovery (HA/DR) solutions meet our Enterprise customer standards for Gold and Platinum MAA solutions.

For the latest details, see Oracle Cloud Maximum Availability Architecture.

The following table outlines MAA validated solutions and guidance for Oracle Database Cloud services.

Service	Bronze	Silver	Gold	Platinum
Oracle Base Database Service (BaseDB)	Base DB – Single Instance • Use Oracle Database cloud automation to configure the network and create DB Systems and databases • Use Oracle Database cloud automation for system and database life cycle operations including software updates, upgrades, monitoring, alerting and database administration and management. • Use cloud-managed backup service. Recommended: Use (Zero Data Loss) Autonomous Recovery Service and real time redo to reduce data loss in case of disasters • Customer's responsibilities – Create test systems to evaluate application, configuration or software changes – Execute System and database sizing, resource management and monitoring	Base DB – 2-Nodes Oracle RAC • Bronze, plus the following: • Use Oracle Database cloud automation to create multi-node RAC DB systems on Oracle Cloud Infrastructure (OCI) • Customer's responsibilities – Maximize application failover and uptime with Achieving Continuous Availability For Your Applications	Base DB – 2-Nodes Oracle RAC with Active Data Guard • Silver, plus the following: • Recommended: primary and standby database systems are symmetric in shape and system resources. • Use cloud automation to create and manage standby database. The location can be in another Availability Domain or Region for better fault isolation. • Customer's responsibilities: – Configure Fast Start Failover (automatic failover) with MAA practices to bound RTO after database, cluster, potentially Availability Domain or regional failure – Tune Data Guard if lagging (see Tune and Troubleshoot Oracle Data Guard – Extend application failover to standby with Achieving Continuous Availability For Your Applications or use Full Stack Disaster Recovery Service.	Base DB – 2-Nodes Oracle RAC with Active Data Guard and Oracle GoldenGate • Gold, plus the following: • Customer's responsibilities: – Set up GoldenGate Cloud: Configuring Oracle GoldenGate Hub for MAA Platinum. GoldenGate management and tuning. – Extend application failover to Oracle GoldenGate replica with Achieving Continuous Availability For Your Applications – Optionally use Global Data Services for smart workload management between GoldenGate replicas and standby databases

Service	Bronze	Silver	Gold	Platinum
Oracle Exadata Database Service on Dedicated Infrastructure (ExaDB-D)	NA	ExaDB-D (Default) • Use Oracle Database cloud automation to create Exadata infrastructure, VM cluster, and RAC databases. • Use Oracle Database cloud automation for system and database life cycle operations including software updates, upgrades, monitoring, service events and health alerts, and database administration and management. • Use cloud-managed backup service. Recommended: Use (Zero Data Loss) Autonomous Recovery Service and real time redo to reduce data loss in case of disasters • Customer Responsibilities: – Create test systems to evaluate application, configuration or software changes. – Execute system and database sizing, resource management and monitoring and reviewing exachk (holistic health) – Maximize application failover and	ExaDB-D with Active Data Guard • Silver, plus the following: • Recommended: primary and standby database systems are symmetric in shape and system resources. • Use cloud automation to create and manage standby database. The location can be in another Availability Domain or Region for better fault isolation. • Customer's responsibilities: – Configure Fast Start Failover (automatic failover) with MAA practices to bound RTO after database, cluster, potentially Availability Domain or regional failure – Tune Data Guard if lagging (see Tune and Troubleshoot Oracle Data Guard – Extend application failover to standby with Achieving Continuous Availability For Your Applications or use Full Stack Disaster Recovery Service.	ExaDB-D with Active Data Guard and Oracle GoldenGate • Gold, plus the following: • Customer's responsibilities: – Set up GoldenGate Cloud: Configuring Oracle GoldenGate Hub for MAA Platinum. GoldenGate management and tuning. – Extend application failover to Oracle GoldenGate replica with Achieving Continuous Availability For Your Applications – Optionally use Global Data Services for smart workload management between GoldenGate replicas and standby databases

Service	Bronze	Silver	Gold	Platinum
			uptime with Achieving Continuous Availability For Your Applications	

Service	Bronze	Silver	Gold	Platinum
Oracle Exadata Database Service on Cloud@Customer (ExaDB-C@C)	NA	ExaDB-CC (Default) • Use Oracle Database cloud automation to create Exadata infrastructure, VM cluster, and RAC databases. • Use Oracle Database cloud automation for system and database life cycle operations including software updates, upgrades, monitoring, service events and health alerts, and database administration and management. • Use cloud-managed backup service. Recommended: Use Zero Data Loss Recovery Server and real time redo to reduce data loss in case of disasters • Customer Responsibilities: – Create test systems to evaluate application, configuration or software changes. – Execute system and database sizing, resource management and monitoring and reviewing exachk (holistic health) – Maximize application failover and uptime with	ExaDB-CC with Active Data Guard • Silver, plus the following: • Recommended: primary and standby database systems are symmetric in shape and system resources. • Use cloud automation to create and manage standby database. The location can be in another Availability Domain or Region for better fault isolation. • Customer's responsibilities: – Configure Fast Start Failover (automatic failover) with MAA practices to bound RTO after database, cluster, potentially Availability Domain or regional failure – Tune Data Guard if lagging (see Tune and Troubleshoot Oracle Data Guard – Extend application failover to standby with Achieving Continuous Availability For Your Applications or use Full Stack Disaster Recovery Service.	ExaDB-CC with Active Data Guard and Oracle GoldenGate • Gold, plus the following: • Customer's responsibilities: – Set up On-Premises: Configuring Oracle GoldenGate Hub – Extend application failover to Oracle GoldenGate replica with Achieving Continuous Availability For Your Applications – Optionally use Global Data Services for smart workload management between GoldenGate replicas and standby databases

Service	Bronze	Silver	Gold	Platinum
		Achieving Continuous Availability For Your Applications		
Autonomous Database Serverless (ADB-S)	NA	ADB-S (Default) • Use Oracle Database cloud automation life cycle operations including software updates, upgrades, monitoring, and database administration and management. • Customer Responsibilities: – Create test systems to evaluate application changes – Maximize application failover and uptime with Achieving Continuous Availability For Your Applications	ADB-S with Autonomous Data Guard • Silver, plus the following: • Enable Autonomous Data Guard, choose automatic failover and data loss tolerance. The location can be in another Availability Domain for better fault isolation. • Customer's responsibility: – Extend application failover to standby with Achieving Continuous Availability For Your Applications or use Full Stack Disaster Recovery Service. MAA evaluation completed only with a standby database within the same region or cross Availability-Domains. Cross-Region Autonomous Data Guard Evaluation In Progress	Planned

Service	Bronze	Silver	Gold	Platinum
Autonomous Database on Dedicated Infrastructure (ADB-D)	NA	ADB-D (Default) • Use Oracle Database cloud automation life cycle operations including software updates, upgrades, monitoring, and database administration and management. • Customer Responsibilities: – Create test systems to evaluate application changes – Maximize application failover and uptime with Achieving Continuous Availability For Your Applications	ADB-D with Autonomous Data Guard • Silver, plus the following: • Enable Autonomous Data Guard, choose protection mode, data loss tolerance, and enable automatic failover. The location can be in another Availability Domain or Region for better fault isolation. • Customer's responsibility: – Extend application failover to standby with Achieving Continuous Availability For Your Applications or use Full Stack Disaster Recovery Service.	ADB-D with Autonomous Data Guard and Oracle GoldenGate • Gold, plus the following: • Customer's responsibilities: – Set up GoldenGate Cloud: Configuring Oracle GoldenGate Hub for MAA Platinum. GoldenGate management and tuning. – Extend application failover to Oracle GoldenGate replica with Achieving Continuous Availability For Your Applications

29

Oracle Maximum Availability Architecture and Oracle Autonomous Database

Oracle Maximum Availability Architecture (MAA) is a set of best practices developed by Oracle engineers over many years for the integrated use of Oracle High Availability, data protection, and disaster recovery technologies.

The key goal of Oracle MAA is to meet Recovery Time Objectives (RTO) and Recovery Point Objectives (RPO) for Oracle databases and applications running on our system and database platforms using Oracle Cloud MAA architectures and solutions.

See Oracle MAA Reference Architectures for an overview of the MAA reference architectures and their associated benefits and potential RTO and RPO targets. Also, see Oracle Maximum Availability Architecture in Oracle Exadata Cloud Systems for the inherent differentiated Oracle Exadata Cloud HA and data protection benefits, because Autonomous Database Cloud runs on the Exadata Cloud platform.

Note that Maximum Availability Architectures leverage Chaos Engineering throughout its testing and development life cycles to ensure that end-to-end application and database availability is preserved, or at its optimal levels, for any fault or maintenance event in Oracle Cloud. *Chaos Engineering* is the discipline of experimenting on a system to build confidence in the system's capability to withstand turbulent conditions in production. Specifically, MAA aggressively injects various faults and planned maintenance events to evaluate application and database impact throughout our development, stress, and testing cycles. With that experimentation, best practices, defects, and lessons learned are derived, and that knowledge is put back into practice to evolve and improve our cloud MAA solutions.

Oracle Autonomous Database with Default High Availability Option (MAA Silver)

High availability is suitable for all development, test, and production databases that have high uptime requirements and zero or low data loss tolerance. By default, Autonomous Databases are highly available, incorporating a multi-node configuration to protect against localized software and hardware failures.

Each Autonomous Database application service resides in at least one Oracle Real Application Clusters (Oracle RAC) instance, with the option to fail over to another available Oracle RAC instance for unplanned outages or planned maintenance activities, enabling zero or near-zero downtime.

Autonomous Database automatic backups are stored in Oracle Cloud Infrastructure Object Storage and are replicated to another availability domain if available. These backups can be used to restore the database in the event of a disaster. For Autonomous Database with Exadata Cloud at Customer, customers have an option to backup to NFS or Zero Data Loss Recovery Appliance (ZDLRA); however, replication of those backups is the responsibility of the customer.

Major database upgrades are automated. For Autonomous Database Serverless, the downtime is minimal.

The uptime service-level agreements (SLAs) per month is 99.95% (a maximum of 22 minutes of downtime per month). To achieve the application uptime SLAs where most months would be zero downtime, see Maintaining Application Uptime below.

The following table describes the recovery-time objectives and recovery-point objectives (data loss tolerance) for different outages.

Table 29-1 Default High Availability Policy Recovery Time (RTO) and Recovery Point (RPO) Service-level Objectives

Failure and Maintenance Events	Database Downtime	Service-level Downtime (RTO)	Potential Service-level Data Loss (RPO)
Localized events, including: • Exadata cluster network topology failures • Storage (disk and flash) failures • Database instance failures • Database server failures • Periodic software and hardware maintenance updates	Zero	Near-zero	Zero
Events that require restoring from backup because the standby database does not exist: • Data corruptions • Full database failures • Complete storage failures • Availability domain (AD) for multi-AD regions	Minutes to hours (without Autonomous Data Guard)	Minutes to hours (without Autonomous Data Guard)	15 minutes for Oracle Autonomous Database on Dedicated Exadata Infrastructure 1 minute for Autonomous Database Serverless (without Autonomous Data Guard)
Events that require non-rolling software updates or database upgrades	Less than 10 minutes for Autonomous Database Serverless Minutes to hour for Autonomous Database on Dedicated Infrastructure (without Autonomous Data Guard)	Less than 10 minutes for Autonomous Database Serverless Minutes to hour for Autonomous Database on Dedicated Infrastructure (without Autonomous Data Guard)	Zero

In the table above, the amount of downtime for events that require restoring from a backup varies depending on the nature of the failure. In the most optimistic case, physical block corruption is detected and the block is repaired with block media recovery in minutes. In this case, only a small portion of the database is affected with zero data loss. In a more pessimistic case, the entire database or cluster fails, then the database is restored and recovered using the latest database backup, including all archives.

Data loss is limited by the last successful archive log backup, the frequency of which is every 15 minutes for Autonomous Database on Dedicated Infrastructure and 1 minute for Autonomous Database Serverless. Archive or redo are backed up to Oracle Cloud Infrastructure Object Storage or File Storage Service for future recovery purposes. Data loss can be seconds, or, at worst minutes of data loss, around the last successful archive log and remaining redo in the online redo logs that were not archived to external storage.

Oracle Autonomous Database with Autonomous Data Guard Option (MAA Gold)

Enable Autonomous Data Guard for mission-critical production databases that require better uptime requirements for disasters from data corruptions, and database or site failures, while still reaping the Autonomous Database High Availability Option benefits.

Additionally, the read-only standby database provides expanded application services to offload reporting, queries, and some updates. The read-only standby database is only available with Autonomous Data Guard on Dedicated Infrastructure.

Enabling Autonomous Data Guard adds one symmetric standby database to an Exadata rack that is located in the same availability domain, another availability domain, or in another region. The primary and standby database systems are configured symmetrically to ensure that performance service levels are maintained after Data Guard role transitions. Autonomous Database Serverless supports configuring two standby databases, and Autonomous Database on Dedicated Infrastructure is restricted to a single database at this time. For Autonomous Database Serverless, a multiple standby configuration consists of a local standby database in the same region and a cross-region standby database.

Oracle Autonomous Data Guard features asynchronous redo transport (in maximum performance mode) by default to ensure zero application performance impact. The standby database can be placed within the same availability domain, across availability domains, or across regions. MAA recommends placing the standby in separate availability domain or in a different region for the best fault isolation. Data Guard zero data loss protection can be achieved by configuring synchronous redo transport (in maximum availability mode); however, maximum availability database protection mode with synchronous redo transport is only available with Autonomous Database on Dedicated Infrastructure, and the standby database is typically placed in a different availability domain in the same region, or across multiple regions if the round trip latency between regions is minimal (< 5ms) to ensure a negligible impact on application response time and throughput while providing fault isolation. Furthermore, local and remote virtual cloud network peering provides a secure, high-bandwidth network across availability domains and regions for any traffic between the primary and standby servers.

Backups are scheduled automatically on both primary and standby databases, and they are stored in Oracle Cloud Infrastructure Object Storage. Autonomous Database with Exadata Cloud at Customer, provides you with an option to backup to NFS or Zero Data Loss Recovery Appliance; however, replication of those backups is the responsibility of the customer. Those backups can be used to restore databases in the event of a double disaster, where both primary and standby databases are lost.

The uptime service-level agreement (SLA) per month is 99.995% (maximum 132 seconds of downtime per month) and recovery time objectives (downtime) and recovery point objectives (data loss) are low, as described in the table below. To achieve the application uptime SLAs where most months would be zero downtime, refer to Maintaining Application Uptime (XREF).

Automatic Data Guard failover with Autonomous Database Serverless supports a data loss threshold service level which will initiate an automatic failover to the standby database if the data loss is below that threshold. Zero data loss failover is not guaranteed for Autonomous Database Serverless but possible when the primary database fails while primary system container and infrastructure is still available allowing the remaining redo to be sent and applied to the standby database. Automatic Data Guard failover with Autonomous Database on Dedicated Infrastructure supports zero data loss or low data loss threshold service levels. In all cases, automatic Autonomous Data Guard failover will occur for primary database, cluster, or data center failures when those data loss service levels can be guaranteed. The target standby

becomes the new primary database, and all application services are enabled automatically. A manual Data Failover option is provided in the OCI Console. For the manual Data Guard failover option, the calculated downtime for the uptime SLA starts with the time to execute the Data Guard failover operation and ends when the new primary service is enabled.

Automatic Data Guard failover with Autonomous Database Serverless supports a data loss threshold service level which initiates an automatic failover to the standby database if the data loss is below that threshold. Zero data loss failover is not guaranteed for Autonomous Database Serverless but is possible when the primary database fails while the primary system container and infrastructure are still available, allowing the remaining redo to be sent and applied to the standby database. Automatic Data Guard failover with Autonomous Database on Dedicated Infrastructure supports zero data loss or low and configurable data loss threshold service levels.

In all cases, automatic Autonomous Data Guard failover occurs for primary database, cluster, or data center failures when those data loss service levels can be guaranteed. The target standby becomes the new primary database, and all application services are enabled automatically. A manual Data Failover option is provided in the OCI Console. For the manual Data Guard failover option, the calculated downtime for the uptime SLA starts with the time to execute the Data Guard failover operation and ends when the new primary service is enabled.

You can choose whether your database failover site is located in the same availability domain, in a different availability domain within the same region, or in a different region, contingent upon application or business requirements and data center availability.

Table 29-2 Autonomous Data Guard Recovery Time (RTO) and Recovery Point (RPO) Service-level Objectives

Failure and Maintenance Events	Service-level Downtime (RTO)[1]	Potential Service-level Data Loss (RPO)
Localized events, including: • Exadata cluster network fabric failures • Storage (disk and flash) failures • Database instance failures • Database server failures • Periodic software and hardware maintenance updates	Zero or Near Zero	Zero

Table 29-2 (Cont.) Autonomous Data Guard Recovery Time (RTO) and Recovery Point (RPO) Service-level Objectives

Failure and Maintenance Events	Service-level Downtime (RTO)[1]	Potential Service-level Data Loss (RPO)
Events that require failover to the standby database using Autonomous Data Guard, including: • Data corruptions (because Data Guard has automatic block repair for physical corruptions[2], a failover operation is required only for logical corruptions or extensive data corruptions) • Full database failures • Complete storage failures • Availability domain or region failures[3]	Few seconds to two minutes[4]	Zero with maximum availability protection mode (uses synchronous redo transport). Most commonly used for intra-region standby databases. This is available for Autonomous Data Guard on Dedicated Infrastructure. Near zero for maximum performance protection mode (uses asynchronous redo transport). Most commonly used for cross-region standby databases. Also used for intra-regional standby databases and to ensure zero application impact. This is applicable for both Autonomous Data Guard on Dedicated Infrastructure and Autonomous Database Serverless. RPO is typically less than 10 seconds. RPO can be impacted by network bandwidth and throughput between primary and standby clusters.

[1] Service-Level Downtime (RTO) excludes detection time that includes multiple heartbeats to ensure the source is indeed inaccessible before initiating an automatic failover.

[2] The Active Data Guard automatic block repair for physical corruptions feature is only available for Autonomous Data Guard on Dedicated Infrastructure.

[3]Regional failure protection is only available if the standby is located across regions.

[4] The back end Autonomous Data Guard role transition timings are much faster than what is indicated by the Cloud Console refresh rates.

Both Autonomous Database on Dedicated Infrastructure and Autonomous Database Serverless have been MAA Gold validated and certified. Autonomous Database on Dedicated Infrastructure was validated with a standby database in the same region, and also with a standby database in a different region, and the above SLAs were met when the standby target was symmetric to the primary. RTO and RPO SLAs were met with redo rates of up to 1000 MB/sec. Autonomous Database Serverless was validated and certified with a standby database in the same region only, and met the above SLAs when the standby target had symmetric resources. RTO and RPO SLAs were met with redo rates up to 300 MB/sec for the entire Container Database (CDB) where the target Autonomous Data Guard pluggable database resides.

Autonomous Database with Autonomous Data Guard Option and Oracle GoldenGate (MAA Platinum)

MAA Platinum with Autonomous Database on Dedicated Infrastructure is configurable. No guaranteed SLAs are provided since the GoldenGate and application failover configuration is manual.

MAA Platinum or Never-Down Architecture, delivers near-zero recovery time objective (RTO, or downtime incurred during an outage) and potentially zero or near zero recover point objective (RPO, or data loss potential).

The MAA Platinum with Autonomous Database on Dedicated Infrastructure ensures:

- RTO = zero or near-zero for all local failures
- RTO = zero or near-zero for disasters, such as database, cluster, or site failures, achieved by redirecting the application to an Autonomous Database with Autonomous Data Guard or Oracle GoldenGate replica
- Zero downtime maintenance for software and hardware updates
- Zero downtime database upgrade or application upgrade by redirecting the application to an upgraded Oracle GoldenGate replica residing in a separate Autonomous Database on Dedicated Infrastructure
- RPO = zero or near-zero data loss, depending on selecting the Oracle Data Guard Maximum Availability or Maximum Performance protection modes with synchronous redo transport in Autonomous Database with Autonomous Data Guard
- Fast re-synchronization and zero or near-zero RPO between Oracle GoldenGate source and target databases after a disaster using Cloud MAA GoldenGate Hub and Oracle GoldenGate best practices
- After any database failure, automatic failover to its standby database occurs automatically using integrated Data Guard Fast-start Failover (FSFO). Subsequently, automatic re-synchronization between Oracle GoldenGate source and target databases resumes from the new primary after a role transition. For synchronous transport, this leads to eventual zero data loss.

Prerequisites:

- Autonomous Database on Dedicated Infrastructure must be running Oracle Database software release 19.20 or later for GoldenGate conflict resolution support
- Autonomous Database with Autonomous Data Guard and automatic failover needs to be configured for fast GoldenGate resynchronization after a disaster
- GoldenGate setup must be done manually according to Cloud MAA best practices
- Application failover to an available GoldenGate replica or a new primary database must be configured. Currently, Global Data Services (GDS) cannot be used with an Autonomous Database in this architecture.

Implementing the MAA Platinum Solution

To achieve an MAA Platinum solution, review and leverage the technical briefs and documentation referenced in the following steps.

1. Review Oracle MAA Platinum Tier for Oracle Exadata to understand MAA Platinum benefits and use cases.
    a. Decide primary database locations based on application needs. The primary database will reside in Autonomous Database on Dedicated Infrastructure.
    b. Decide standby database location based on fault isolation requirements.
    c. Enable Autonomous Data Guard.
    d. Choose Autonomous Data Guard protection mode based on RPO tolerance, and set up automatic failover.

2. Set up MAA GoldenGate Hub in Oracle cloud.
   a. Follow the steps in Cloud: Configuring Oracle GoldenGate Hub for MAA Platinum.
   b. Configure Bidirectional Replication and Automatic Conflict Detection and Resolution. See Set Up Bidirectional Replication for Oracle GoldenGate Microservices Architecture or the latest Oracle GoldenGate 21c documentation.
3. Configure application failover options so that your application can fail over automatically in the case of database, cluster, or site failure.

Maintaining Application Uptime

Ensure that network connectivity to Oracle Cloud Infrastructure is reliable so that you can access your tenancy's Autonomous Database resources.

Follow the guidelines to connect to your Autonomous Database (see Autonomous Database Serverless, or Autonomous Database on Dedicated Exadata Infrastructure). Applications must connect to the predefined service name and download client credentials that include the proper `tnsnsames.ora` and `sqlnet.ora` files. You can also change your specific application service's `drain_timeout` attribute to fit your requirements.

For more details about enabling continuous application service through planned and unplanned outages, see Configuring Continuous Availability for Applications. Oracle recommends that you test your application readiness by following Validating Application Failover Readiness (Doc ID 2758734.1).

For Oracle Exadata Cloud Infrastructure planned maintenance events that require restarting database instance, Oracle automatically relocates services and drain sessions to another available Oracle RAC instance before stopping any Oracle RAC instance. For OLTP applications that follow the MAA checklist, draining and relocating services results in zero application downtime.

Some applications, such as long running batch jobs or reports, may not be able to drain and relocate gracefully, even with a longer drain timeout. For those applications, Oracle recommends that you schedule the software planned maintenance window excluding these types of activities, or stop these activities before the planned maintenance window. For example, you can reschedule a planned maintenance window so that it is outside your batch windows, or stop batch jobs before a planned maintenance window.

30

Oracle Maximum Availability Architecture in Oracle Exadata Cloud Systems

Oracle Maximum Availability Architecture in Oracle Exadata Cloud Infrastructure (ExaDB-D) and Oracle Exadata Cloud@Customer (ExaDB-C@C) provides inherent high availability, data protection, and disaster recovery protection integrated with both cloud automation and life cycle operations, enabling Oracle Exadata Cloud systems to be the best cloud solution for enterprise databases and applications.

See Oracle Cloud: Maximum Availability Architecture for detailed walk-through of Oracle Cloud MAA architectures and features.

Oracle Maximum Availability Architecture Benefits

- **Deployment**: Oracle Exadata Cloud systems (ExaDB-D and ExaDB-C@C)) are deployed using Oracle Maximum Availability Architecture best practices, including configuration best practices for storage, network, operating system, Oracle Grid Infrastructure, and Oracle Database. ExaDB-D is optimized to run enterprise Oracle databases with extreme scalability, availability, and elasticity.
- **Oracle Maximum Availability Architecture database templates**: All Oracle Cloud databases created with Oracle Cloud automation use Oracle Maximum Availability Architecture default settings, which are optimized for ExaDB-D.

  Oracle does not recommend that you use custom scripts to create cloud databases. Other than adjusting memory and system resource settings, avoid migrating previous database parameter settings, especially undocumented parameters. One beneficial database data protection parameter, `DB_BLOCK_CHECKING`, is not enabled by default due to its potential overhead. MAA recommends evaluating the performance impact for your application and enabling this setting if performance impact is reasonable.
- **Backup and restore automation**: When you configure automatic backup to Oracle Cloud Infrastructure Object Storage, backup copies provide additional protection when multiple availability domains exist in your region, and RMAN validates cloud database backups for any physical corruptions.

  Database backups occur daily, with a full backup occurring once per week and incremental backups occurring on all other days. Archive log backups occur frequently to reduce potential data loss in case of disaster. The archive log frequency is typically 30 minutes.
- **Oracle Exadata Database Machine inherent benefits**: Oracle Exadata Database Machine is the best Oracle Maximum Availability Architecture platform that Oracle offers. Exadata is engineered with hardware, software, database, and availability innovations that support the most mission-critical enterprise applications.

  Specifically, Exadata provides unique high availability, data protection, and quality-of-service capabilities that set Oracle apart from any other platform or cloud vendor. Sizing Exadata cloud systems to meet your application and database system resource needs (for example, sufficient CPU, memory, and I/O resources) is very important to maintain the highest availability, stability, and performance. Proper sizing is especially important when consolidating many databases on the same cluster.

For a comprehensive list of Oracle Maximum Availability Architecture benefits for Oracle Exadata Database Machine systems, see Exadata Database Machine: Maximum Availability Architecture Best Practices.

Examples of these benefits include:

- **High availability and low brownout**: Fully-redundant, fault-tolerant hardware exists in the storage, network, and database servers. Resilient, highly-available software, such as Oracle Real Application Clusters (Oracle RAC), Oracle Clusterware, Oracle Database, Oracle Automatic Storage Management, Oracle Linux, and Oracle Exadata Storage Server enable applications to maintain application service levels through unplanned outages and planned maintenance events.

  For example, Exadata has instant failure detection that can detect and repair database node, storage server, and network failures in less than two seconds, and resume application and database service uptime and performance. Other platforms can experience 30 seconds, or even minutes, of blackout and extended application brownouts for the same type of failures. Only the Exadata platform offers a wide range of unplanned outage and planned maintenance tests to evaluate end-to-end application and database brownouts and blackouts.

- **Data protection**: Exadata provides Oracle Database with physical and logical block corruption prevention, detection, and, in some cases, automatic remediation.

  The Exadata Hardware Assisted Resilient Data (HARD) checks include support for server parameter files, control files, log files, Oracle data files, and Oracle Data Guard broker files, when those files are stored in Exadata storage. This intelligent Exadata storage validation stops corrupted data from being written to disk when a HARD check fails, which eliminates a large class of failures that the database industry had previously been unable to prevent.

  Examples of the Exadata HARD checks include:

  – Redo and block checksum
  – Correct log sequence
  – Block type validation
  – Block number validation
  – Oracle data structures, such as block magic number, block size, sequence number, and block header and tail data structures

  Exadata HARD checks are initiated from Exadata storage software (cell services) and work transparently after enabling a database `DB_BLOCK_CHECKSUM` parameter, which is enabled by default in the cloud. Exadata is the only platform that currently supports the HARD initiative.

  Furthermore, Oracle Exadata Storage Server provides non-intrusive, automatic hard disk scrub and repair. This feature periodically inspects and repairs hard disks during idle time. If bad sectors are detected on a hard disk, then Oracle Exadata Storage Server automatically sends a request to Oracle Automatic Storage Management (ASM) to repair the bad sectors by reading the data from another mirror copy.

  Finally, Exadata and Oracle ASM can detect corruptions as data blocks are read into the buffer cache, and automatically repair data corruption with a good copy of the data block on a subsequent database write. This inherent intelligent data protection makes Exadata Database Machine and ExaDB-D the best data protection storage platform for Oracle databases.

  For comprehensive data protection, a Maximum Availability Architecture best practice is to use a standby database on a separate Exadata instance to detect, prevent, and

automatically repair corruptions that cannot be addressed by Exadata alone. The standby database also minimizes downtime and data loss for disasters that result from site, cluster, and database failures.

- **Response time quality of service**: Only Exadata has end-to-end quality-of-service capabilities to ensure that response time remains low and optimum. Database server I/O latency capping and Exadata storage I/O latency capping ensure that read or write I/O can be redirected to partnered cells when response time exceeds a certain threshold.

  If storage becomes unreliable (but not failed) because of poor and unpredictable performance, then the disk or flash cache can be confined offline, and later brought back online if heuristics show that I/O performance is back to acceptable levels. Resource management can help prioritize key database network or I/O functionality, so that your application and database perform at an optimized level.

  For example, database log writes get priority over backup requests on Exadata network and storage. Furthermore, rapid response time is maintained during storage software updates by ensuring that partner flash cache is warmed so flash misses are minimized.

- **End-to-end testing and holistic health checks**: Because Oracle owns the entire Oracle Exadata Cloud Infrastructure, end-to-end testing and optimizations benefit every Exadata customer around the world, whether hosted on-premises or in the cloud. Validated optimizations and fixes required to run any mission-critical system are uniformly applied after rigorous testing. Health checks are designed to evaluate the entire stack.

  The Exadata health check utility EXACHK is Exadata cloud-aware and highlights any configuration and software alerts that may have occurred because of customer changes. No other cloud platform currently has this kind of end-to-end health check available. For Oracle Autonomous Database, EXACHK runs automatically to evaluate Maximum Availability Architecture compliance. For non-autonomous databases, Oracle recommends running EXACHK at least once a month, and before and after any software updates, to evaluate any new best practices and alerts.

- **Higher Uptime**: The uptime service-level agreement per month is 99.95% (a maximum of 22 minutes of downtime per month), but when you use MAA best practices for continuous service, most months would have zero downtime.

Full list of Exadata features and benefits: Whats New in Oracle Exadata Database Machine

Oracle Maximum Availability Architecture best practices paper: Oracle Maximum Availability Architecture (MAA) engineering collaborates with Oracle Cloud teams to integrate Oracle MAA practices that are optimized for Oracle Cloud Infrastructure and security. See MAA Best Practices for the Oracle Cloud for additional information about continuous availability, Oracle Data Guard, Hybrid Data Guard, Oracle GoldenGate, and other Maximum Availability Architecture-related topics.

Expected Impact with Unplanned Outages

The following table lists various unplanned outages and the associated potential database downtime, application level Recovery Time Objective (RTO), and data loss potential or recovery point objective (RPO). For Oracle Data Guard architectures, the database downtime or service level downtime does not include detection time or the time it takes before a customer initiates the Cloud Console Data Guard failover operation.

Table 30-1 Availability and Performance Impact for Exadata Cloud Software Updates

Failure and Maintenance Events	Database Downtime	Service-Level Downtime (RTO)	Potential Service-Level Data Loss (RPO)
Localized events, including: Exadata cluster network topology failures Storage (disk and flash) failures Database instance failures Database server failures	Zero	Near-zero	Zero
Events that require restoring from backup because a standby database does not exist: Data corruptions Full database failures Complete storage failures Availability domain	Minutes to hours (without Data Guard)	Minutes to hours (without Data Guard)	30 minutes (without Data Guard)
Events using Data Guard to fail over: Data corruptions Full database failures Complete storage failures Availability domain or region failures	Seconds to minutes[1] Zero downtime for physical corruptions due to auto-block repair feature	Seconds to minutes[1] The foreground process that detects the physical corruption pauses while auto block repair completes	Zero for Max Availability (SYNC) Near Zero for Max Performance (ASYNC)

[1] To protect from regional failure, you will need a standby database in a different region than the primary database.

Expected Impact with Planned Maintenance

The following table lists various software updates and the associated database and application impact. This is applicable for all Oracle Exadata Cloud infrastructures, including Oracle Exadata Cloud@Customer (ExaDB-C@C), Oracle Exadata Cloud Infrastructure (ExaDB-D) Gen2, and Oracle Autonomous Database (ADB).

Table 30-2 Availability and Performance Impact for Oracle Exadata Cloud Software Updates

Software Update	Database Impact	Application Impact	Scheduled By	Performed By
Exadata Network Fabric Switches	Zero downtime with No Database Restart	Zero to single-digit seconds brownout	Oracle schedules based on customer preferences and customer can reschedule	Oracle Cloud for both ADB and non-ADB

Table 30-2 (Cont.) Availability and Performance Impact for Oracle Exadata Cloud Software Updates

Software Update	Database Impact	Application Impact	Scheduled By	Performed By
Exadata Storage Servers	Zero downtime with No Database Restart	Zero to single-digit seconds brownout Exadata storage servers are updated in rolling manner maintaining redundancy Oracle Exadata System Software pre-fetches the secondary mirrors of the OLTP data that is most frequently accessed into the flash cache, maintaining application performance during storage server restarts Exadata smart flash for database buffers is maintained across storage server restart With Exadata 21.2 software, Persistent Storage Index and Persistent Columnar Cache features enable consistent query performance after a storage server software update	Oracle schedules based on customer preferences and customer can reschedule	Oracle Cloud for both ADB and non-ADB
Exadata Database Host - Monthly Infrastructure Security Maintenance	Zero downtime with No Host or Database Restart	Zero downtime	Oracle schedules and customer can reschedule	Oracle Cloud for both ADB and non-ADB
Exadata Database Host - Quarterly Infrastructure Maintenance	Zero downtime with Oracle RAC rolling updates	Zero downtime Exadata Database compute resources are reduced until planned maintenance completes	Oracle schedules based on customer preferences and customer can reschedule	Oracle Cloud for both ADB and non-ADB
Exadata Database Guest	Zero downtime with Oracle RAC rolling updates	Zero downtime Exadata Database compute resources are reduced until planned maintenance completes	Customer for ADB	Oracle Cloud for ADB Customer using Oracle Cloud Console/APIs for non-ADB

Table 30-2 (Cont.) Availability and Performance Impact for Oracle Exadata Cloud Software Updates

Software Update	Database Impact	Application Impact	Scheduled By	Performed By
Oracle Database quarterly update or custom image update	Zero downtime with Oracle RAC rolling updates	Zero downtime Exadata Database compute resources are reduced until planned maintenance completes Special consideration is required during rolling database quarterly updates for applications that use database OJVM. See MOS Note 2217053.1 for details.	Customer for ADB	Oracle Cloud for ADB. For ADB-D, standby-first patch practices are automatically applied. Customer using Oracle Cloud Console/APIs or dbaascli utility for non-ADB. In-place via database home patch, and out-of-place via database move, software updates exist. Works for Data Guard and standby databases (refer to MOS 2701789.1)
Oracle Grid Infrastructure quarterly update or upgrade	Zero downtime with Oracle RAC rolling updates	Zero downtime Exadata Database compute resources are reduced until planned maintenance completes	Customer for ADB	Oracle Cloud for ADB Customer using Oracle Cloud Console/APIs or dbaascli utility for non-ADB
Oracle Database upgrade with downtime	Minutes to Hour(s) downtime	Minutes to Hour(s) downtime	Customer for ADB	Oracle Cloud for ADB Customer using Oracle Cloud Console/APIs or dbaascli utility for non-ADB Works for Data Guard and standby databases (refer to MOS 2628228.1)
Oracle Database upgrade with near zero downtime	Minimal downtime with DBMS_ROLLING, Oracle GoldenGate replication, or with pluggable database relocate	Minimal downtime with DBMS_ROLLING, Oracle GoldenGate replication, or with pluggable database relocate	Customer for non-ADB	Oracle Cloud for ADB on Shared Exadata Infrastructure (ADB-S) can run pluggable database relocate for upgrade use cases Customer using dbaascli for non-autonomous leveraging DBMS_ROLLING. Refer to Exadata Cloud Database 19c Rolling Upgrade With DBMS_ROLLING (Doc ID 2832235.1) Customer using generic Maximum Availability Architecture best practices for non-ADB

Exadata cloud systems have many elastic capabilities that can be used to adjust database and application performance needs. By rearranging resources on need, you can maximize system resources to targeted databases and applications and you can minimize costs. The following table lists elastic Oracle Exadata Cloud Infrastructure and VM Cluster updates, and the impacts associated with those updates on databases and applications. All of these operations can be performed using Oracle Cloud Console or APIs unless specified otherwise.

Table 30-3 Availability and Performance Impact for Exadata Elastic Operations

VM Cluster Changes	Database Impact	Application Impact
Scale Up or Down VM Cluster Memory	Zero downtime with Oracle RAC rolling updates	Zero to single-digit seconds brownout
Scale Up or Down VM Cluster CPU	Zero downtime with No Database Restart	Zero downtime Application performance and throughput can be impacted by available CPU resources
Scale Up or Down (resize) ASM Storage for Database usage	Zero downtime with No Database Restart	Zero downtime Application performance might be minimally impacted.
Scale Up VM Local /u02 File System Size (Exadata X8M and later systems)	Zero downtime with No Database Restart	Zero downtime
Scale Up VM Local /u02 File System Size (Exadata X8 and earlier systems)	Zero downtime with Oracle RAC rolling updates	Zero to single-digit seconds brownout
Scale Down VM Local /u02 File System Size	Zero downtime with Oracle RAC rolling updates for scaling down	Zero to single-digit seconds brownout
Adding Exadata Storage Cells	Zero downtime with No Database Restart	Zero to single-digit seconds brownout Application performance might be minimally impacted
Adding Exadata Database Servers	Zero downtime with No Database Restart	Zero to single-digit seconds brownout Application performance and throughput may increase by adding Oracle RAC instances and CPU resources
Adding/Dropping Database Nodes in Virtual Machines (VMs) Cluster	Zero downtime with No Database Restart	Zero to single-digit seconds brownout Application performance and throughput may increase or decrease by adding or dropping Oracle RAC instances and CPU resources

Because some of these elastic changes may take significant time, and may impact available resources for your application, some planning is required.

Note that “scale down” and “drop” changes will decrease available resources. Care must be taken to not reduce resources below the amount required for database and application stability and to meet application performance targets. Refer to the following table for estimated timings and planning recommendations.

Table 30-4 Customer Planning Recommendations for Exadata Elastic Operations

VM Cluster Changes	Estimated Timings	Customer Planning Recommendations
Scale Up or Down VM Cluster Memory	Time to drain services and Oracle RAC rolling restart Typically 15-30 minutes per node, but may vary depending on application draining	Understanding application draining. See Achieving Continuous Availability For Your Applications Before scaling down memory, ensure that database SGAs can still be stored in hugepages, and that application performance is still acceptable. To preserve predictable application performance and stability: • Monitor and scale up before important high workload patterns require the memory resources • Avoid memory scale down unless all your Databases' SGA and PGA memory fit into the new memory size and that all SGAs are accommodated by system's hugepages.
Scale Up or Down VM Cluster CPU	Online operation, typically less than 5 minutes per VM cluster. Scaling up from a very low value to very high value (10+ oCPU increase) may take 10 minutes.	To preserve predictable application performance and stability: • Monitor and scale up before important high workload patterns require the CPU resources or when consistently reaching an OCPU threshold for tolerated amount of time. • Only scale down if the load average is below a threshold for at least 30 minutes or scale down based on fixed workload schedules (e.g. business hours with 60 OCPUs, non-business hours with 10 OCPUs and batch with 100 oCPUs) • Avoid more than one scale down requests within 2 hours period

Table 30-4 (Cont.) Customer Planning Recommendations for Exadata Elastic Operations

VM Cluster Changes	Estimated Timings	Customer Planning Recommendations
Scale Up or Down (resize) ASM Storage for Database usage	Time varies based on utilized database storage capacity and database activity. The higher percentage of utilized database storage, the longer the resize operation (which includes ASM rebalance) will take. Typically minutes to hours.	Oracle ASM rebalance is initiated automatically. Storage redundancy is retained. Due to inherent best practices of using non-intrusive ASM power limit, application workload impact is minimal. Choose a non-peak window so resize and rebalance operations can be optimized. Since the time may vary significantly, plan for the operation to complete in hours. To estimate the time that an existing resize or rebalance operation per VM cluster, query `GV$ASM_OPERATION`. For example, a customer can run the following query every 30 minutes to evaluate how much work (EST_WORK) and how much more time (EST_MINUTES) potentially is required: `select operation, pass, state, sofar, est_work, est_minutes from gv$asm_operation where operation='REBAL';` Note the estimated statistics tend to become more accurate as the rebalance progresses but can vary based on the concurrent workload.
Scale Up VM Local /u02 File System Size (Exadata X8M and later)	Online operation, typically less than 5 minutes per VM cluster	VM local file system space is allocated on local database host disks, which is shared by all VM guests for all VM clusters provisioned on that database host. Do not scale up space for Local /u02 File System unnecessarily on one VM cluster such that no space remains to scale up on other VM clusters on the same Exadata Infrastructure because Local /u02 File System scale down must be performed in a RAC rolling manner, which may cause application disruption.
Scale Up VM Local /u02 File System Size (Exadata X8 and earlier)	Time to drain services and Oracle RAC rolling restart. Typically 15-30 minutes per node, but may vary depending on application draining settings.	Understanding application draining. See Achieving Continuous Availability For Your Applications
Scale Down VM Local /u02 File System Size	Time to drain services and Oracle RAC rolling restart. Typically 15-30 minutes per node, but may vary depending on application draining settings.	Understanding application draining See Achieving Continuous Availability For Your Applications

Table 30-4 (Cont.) Customer Planning Recommendations for Exadata Elastic Operations

VM Cluster Changes	Estimated Timings	Customer Planning Recommendations
Adding Exadata Storage Cells	Online operation to create more available space for administrator to choose how to distribute. Typically 3-72 hours per operation depending number of VM clusters, database storage usage and storage activity. With very active database and heavy storage activity, this can take up to take 72 hours. As part of the add storage cell operation, there are two parts to this operation. 1) storage is added to the system as part the add storage, 2) administrator needs to decide which VM cluster to expand its ASM disk groups as a separate operation.	Plan to add storage when your storage capacity utilization will hit 80% within a month's time since the operation may complete in days. Oracle ASM rebalance is initiated automatically. Storage redundancy is retained. Due to inherent best practices of using non-intrusive ASM power limit, application workload impact is minimal. Since the time may vary significantly, plan for the operation to complete in days before the storage is available. To estimate the time that an existing resize or rebalance operation per VM cluster, query `GV$ASM_OPERATION`. For example a customer can run the following query every 30 minutes to evaluate how much work (EST_WORK) and how much more time (EST_MINUTES) potentially is required: `select operation, pass, state, sofar, est_work, est_minutes from gv$asm_operation where operation='REBAL';` Note the estimated statistics tend to become more accurate as the rebalance progresses, but can vary based on the concurrent workload.
Adding Exadata Database Servers	Online operation to expand your VM cluster. One step process to add the Database Compute to the ExaDB-D and then expand the VM cluster. Approximately 1 to 6 hours per Exadata Database Server	Plan to add Database Compute when your Database resource utilization will hit 80% within a month's time. Be aware and plan for this operation to take many hours to a day. Choose a non-peak window so that the add Database Compute operation can complete faster Each Oracle RAC database registered by Oracle Clusterware and visible in the Oracle Cloud Console is extended. If a database was configured outside the Oracle Cloud Console or without dbaascli, then those databases will not be extended.
Adding/Dropping Database Nodes in Virtual Machines (VMs) Cluster	Zero database downtime when adding Database Nodes in VM cluster typically takes 3-6 hours, depending on the number of databases in the VM cluster Zero database downtime with dropping Database Nodes in VM cluster typically takes 1-2 hours, depending on number of databases in the VM cluster	Understand that the add/drop operation is not instantaneous, and operation may take several hours to complete Drop operation reduces Database compute, OCPU and memory resources, so application performance can be impacted

Achieving Continuous Availability For Your Applications

As part of Oracle Exadata Database Service (ExaDB-D and ExaDB-C@C) all software updates (except for non-rolling database upgrades or non-rolling patches) can be done online or with Oracle RAC rolling updates to achieve continuous database up time. Furthermore, any local failures of storage, Exadata network, or Exadata database server are managed automatically, and database up time is maintained.

To achieve continuous application up time during Oracle RAC switchover or failover events, follow these application-configuration best practices:

- Use Oracle Clusterware-managed database services to connect your application. For Oracle Data Guard environments, use role based services.
- Use recommended connection string with built-in timeouts, retries, and delays, so that incoming connections do not see errors during outages.
- Configure your connections with Fast Application Notification.
- Drain and relocate services. Refer to the table below and use recommended best practices that support draining, such as test connections, when borrowing or starting batches of work, and return connections to pools between uses.
- Leverage Application Continuity or Transparent Application Continuity to replay in-flight uncommitted transactions transparently after failures.

For more details on the above checklist, refer to Configuring Continuous Availability for Applications. Oracle recommends testing your application readiness by following Validating Application Failover Readiness (Doc ID 2758734.1).

Depending on the Oracle Exadata Database Service planned maintenance event, Oracle attempts to automatically drain and relocate database services before stopping any Oracle RAC instance. For OLTP applications, draining and relocating services typically work very well and result in zero application downtime.

Some applications, such as long running batch jobs or reports, may not be able to drain and relocate gracefully within the maximum draining time. For those applications, Oracle recommends scheduling the software planned maintenance window around these types of activities or stopping these activities before the planned maintenance window. For example, you can reschedule a planned maintenance window to run outside your batch windows, or stop batch jobs before a planned maintenance window.

Special consideration is required during rolling database quarterly updates for applications that use database OJVM. See MOS Note 2217053.1 for details.

The following table lists planned maintenance events that perform Oracle RAC instance rolling restart, and the relevant service drain timeout variables that may impact your application.

Table 30-5 Application Drain Attributes for Exadata Cloud Software Updates and Elastic Operations

Oracle Exadata Database Service Software Updates or Elastic Operations	Drain Timeout Variables
Oracle DBHOME patch apply and database MOVE	Oracle Cloud software automation stops/relocates database services while honoring drain_timeout settings defined by database service configuration (for example, srvctl).[1] You can override drain_timeout defined on services by using option `–drainTimeoutInSeconds` with command line operation `dbaascli dbHome patch` or `dbaascli database move`. The Oracle Cloud internal maximum draining time supported is 2 hours.
Oracle Grid Infrastructure (GI) patch apply and upgrade	Oracle Cloud software automation stops/relocates database services while honoring drain_timeout settings defined by database service configuration (for example,. srvctl).1 You can override `drain_timeout` defined on services by using option `–drainTimeoutInSeconds` with command line operation `dbaascli grid patch` or `dbaascli grid upgrade`. The Oracle cloud internal maximum draining time supported is 2 hours.
Virtual machine operating system software update (Exadata Database Guest)	Exadata patchmgr/dbnodeupdate software program calls drain orchestration (rhphelper). Drain orchestration has the following drain timeout settings (See Using RHPhelper to Minimize Downtime During Planned Maintenance on Exadata (Doc ID 2385790.1) for details): • DRAIN_TIMEOUT – if a service does not have drain_timeout defined, then this value is used. Default value is 180 seconds. • MAX_DRAIN_TIMEOUT - overrides any higher drain_timeout value defined by database service configuration. Default value is 300 seconds. There is no maximum value. DRAIN_TIMEOUT settings defined by database service configuration are honored during service stop/relocate.

Table 30-5 (Cont.) Application Drain Attributes for Exadata Cloud Software Updates and Elastic Operations

Oracle Exadata Database Service Software Updates or Elastic Operations	Drain Timeout Variables
Exadata X8 and earlier systems • Scale up and down VM local /u02 file system size • Scale up or down VM cluster memory	Exadata X8 and earlier systems local file system resize operation calls drain orchestration (rhphelper). Drain orchestration has the following drain timeout settings (See Using RHPhelper to Minimize Downtime During Planned Maintenance on Exadata (Doc ID 2385790.1) for details): • DRAIN_TIMEOUT – if a service does not have drain_timeout defined, then this value is used. Default value is 180 seconds. • MAX_DRAIN_TIMEOUT - overrides any higher drain_timeout value defined by database service configuration. Default value is 300 seconds. DRAIN_TIMEOUT settings defined by database service configuration are honored during service stop/relocate. The Oracle Cloud internal maximum draining time supported for this operation is 300 seconds.
Exadata X8M and later systems • Scale down VM local file system size	Exadata X8M and later systems call drain orchestration (rhphelper). Drain orchestration has the following drain timeout settings (See Using RHPhelper to Minimize Downtime During Planned Maintenance on Exadata (Doc ID 2385790.1) for details): • DRAIN_TIMEOUT – if a service does not have drain_timeout defined, then this value is used. Default value is 180 seconds. • MAX_DRAIN_TIMEOUT - overrides any higher drain_timeout value defined by database service configuration. Default value is 300 seconds. DRAIN_TIMEOUT settings defined by database service configuration are honored during service stop/relocate. The Oracle Cloud internal maximum draining time supported for this operation is 300 seconds.

Table 30-5 (Cont.) Application Drain Attributes for Exadata Cloud Software Updates and Elastic Operations

Oracle Exadata Database Service Software Updates or Elastic Operations	Drain Timeout Variables
Exadata X8M and later systems • Scale up or down VM cluster memory	Exadata X8M and later systems call drain orchestration (rhphelper). Drain orchestration has the following drain timeout settings (See Using RHPhelper to Minimize Downtime During Planned Maintenance on Exadata (Doc ID 2385790.1) for details): • DRAIN_TIMEOUT – if a service does not have drain_timeout defined, then this value is used. Default value is 180 seconds. • MAX_DRAIN_TIMEOUT - overrides any higher drain_timeout value defined for a given service, default 300. DRAIN_TIMEOUT settings defined by database service configuration are honored during service stop/relocate. The Oracle Cloud internal maximum draining time supported for this operation is 300 seconds.

Table 30-5 (Cont.) Application Drain Attributes for Exadata Cloud Software Updates and Elastic Operations

Oracle Exadata Database Service Software Updates or Elastic Operations	Drain Timeout Variables
Oracle Exadata Cloud Infrastructure (ExaDB-D) software update	The ExaDB-D database host calls drain orchestration (rhphelper). Drain orchestration has the following drain timeout settings (See Using RHPhelper to Minimize Downtime During Planned Maintenance on Exadata (Doc ID 2385790.1) for details): • DRAIN_TIMEOUT – if a service does not have drain_timeout defined, then this value is used. Default value is 180 seconds. • MAX_DRAIN_TIMEOUT - overrides any higher drain_timeout value defined by database service configuration. Default value is 300 seconds. DRAIN_TIMEOUT settings defined by database service configuration are honored during service stop/relocate. The Oracle Cloud internal maximum draining time supported for this operation is • For Exadata X8 and earlier systems, the timeout is 300 seconds. • For Exadata X8M and later systems, the timeout is 500 seconds. Enhanced Infrastructure Maintenance Controls feature: To achieve draining time longer than the Oracle Cloud internal maximum, leverage the custom action capability of the Enhanced Infrastructure Maintenance Controls feature, which allows you to suspend infrastructure maintenance before the next database server update starts, then directly stop/relocate database services running on the database server, and then resume infrastructure maintenance to proceed to the next database server. This feature is also currently available for Oracle Exadata Cloud@Customer (ExaDB-C@C). See Configure Oracle-Managed Infrastructure Maintenance in Oracle Cloud Infrastructure Documentation for details.

[1] Minimum software requirements to achieve this service drain capability is 1) Oracle Database 12.2 and later and 2) the latest Oracle Cloud DBaaS tooling software

Oracle Maximum Availability Architecture Reference Architectures in Oracle Exadata Cloud

Oracle Exadata Cloud (ExaDB-D and ExaDB-C@C) supports all Oracle Maximum Availability Architecture reference architectures, providing support for all Oracle Databases, regardless of their specific high availability, data protection, and disaster recovery service-level agreements.

See MAA Best Practices for the Oracle Cloud for more information about Oracle Maximum Availability Architecture in the Oracle Exadata Cloud.

31

Oracle Maximum Availability Architecture for Oracle Database@Azure

Oracle Maximum Availability Architecture (MAA) in Oracle Exadata Database Service on Dedicated Infrastructure (ExaDB-D) running within Microsoft Azure's data centers ensures inherent high availability, including zero database downtime for software updates and elastic operations.

When augmented with an Oracle Cloud standby database with Oracle Active Data Guard, this cloud MAA architecture achieves comprehensive data protection and disaster recovery. This integrated combination of optimized Exadata hardware, Exadata Cloud software automation, and Oracle MAA best practices enables Oracle Exadata Cloud systems to be the best cloud solution for mission-critical enterprise databases and applications.

As of now, the Oracle MAA solution team has validated and certified the MAA Silver and Gold service level reference architectures with Oracle Database@Azure within the same Azure Region when configured with primary and standby databases residing on Oracle Database@Azure in different Availability Zones (AZ).

See Oracle Cloud: Maximum Availability Architecture for a detailed walk-through of Oracle Cloud MAA architectures and features.

Oracle Database@Azure Evaluations by Oracle MAA

Oracle MAA has evaluated and endorsed Oracle Database@Azure for the MAA Silver architecture on ExaDB-D, and MAA Gold when the standby database resides in another ExaDB-D in separate Availability Zones (AZ).

To ensure success and consistency for Oracle customers, the Oracle MAA team conducts ongoing evaluations of MAA reference architectures on Oracle Database@Azure. MAA solutions based on these evaluations protect your database from outages such as instance, node, storage, network, various data corruptions, and Azure AZ failures, while enabling zero database downtime during software updates, elastic configuration changes, or storage and compute additions.

What Does Oracle MAA Evaluate

An MAA evaluation of Oracle Database@Azure consists of:

- Cloud setup of MAA Silver and MAA Gold architectures in Oracle Database@Azure AZs
- Application throughput and response time impact analysis while injecting 100+ outages (Oracle MAA chaos evaluation)
- Backup and restore performance, throughput, and key use cases
- Oracle Data Guard role transition performance and timings for disaster recovery use cases
- Application impact on elastic ExaDB-D cluster operations
- Application impact on software updates to the ExaDB-D targets
- Data center failure analysis

MAA Silver

MAA Silver on Oracle Database@Azure consists of the following architecture:

- The ExaDB-D cluster residing in Azure hosts one or more databases
- High Availability (HA) and redundant application tier spread across multiple AZs
- Key Management Service and Object Storage Service (for backup and restore) are located on Oracle Cloud Infrastructure (OCI)
- Pre-configured redundant and HA network topology

MAA Gold

MAA Gold on Oracle Database@Azure consists of the following architecture:

- ExaDB-D clusters (primary and standby databases) residing in separate Azure Availability Zones (AZ). Note that all primary and standby databases and their data reside in Oracle Database@Azure. If primary and standby databases reside in the same AZ, this MAA Gold architecture still provides inherent HA benefits plus DR failover options for database and cluster failures, but lacks DR protection for a complete AZ failure.
- HA and redundant application tier spread across multiple AZs
- Key Management Service and Object Storage Service (for backup and restore) are located on Oracle Cloud Infrastructure (OCI)
- Pre-configured redundant and HA network topology

Oracle Maximum Availability Architecture Benefits

The following are some of the benefits of implementing Oracle MAA reference architectures for Oracle Database@Azure.

For a comprehensive list of Oracle Maximum Availability Architecture benefits for Oracle Exadata Database Machine systems, see Exadata Database Machine: Maximum Availability Architecture.

Deployment

Oracle Database@Azure running Oracle Exadata Database Service on Dedicated Infrastructure is deployed using Oracle Maximum Availability Architecture best practices, including configuration best practices for storage, network, operating system, Oracle Grid Infrastructure, and Oracle Database. ExaDB-D is optimized to run enterprise Oracle databases with extreme scalability, availability, and elasticity.

Oracle MAA Database Templates

All Oracle Cloud databases created with Oracle Cloud automation use Oracle Maximum Availability Architecture default settings, which are optimized for Oracle Database@Azure. Oracle does not recommend that you use custom scripts to create cloud databases.

Other than adjusting memory and system resource settings, avoid migrating previous database parameter settings, especially undocumented parameters. One beneficial primary database data protection parameter, `DB_BLOCK_CHECKING`, is not enabled by default due to its potential performance overhead. Any Oracle standby database configured with cloud automation will enable `DB_BLOCK_CHECKING` on the standby automatically to maximize data protection and detection on the standby database. MAA recommends evaluating the performance impact of your application and allowing this setting to be on the primary database to maximize logical

data corruption prevention and detection if the performance impact is reasonable. In Oracle Database versions 19c and later, the Data Guard broker will maintain the data protection settings through MAA best practices.

Backup and Restore Automation

When you configure automatic backup to Oracle Cloud Infrastructure Object Storage, backup copies provide additional protection when multiple availability zones exist in your region. Oracle Recovery Manager (RMAN) validates cloud database backups for any physical corruptions.

Database backups occur daily, with a full backup occurring once per week and incremental backups occurring on all other days. Archived log backups occur frequently to reduce potential data loss in case of full database restore and recovery is required. The archived log backup frequency is 30 minutes by default; however, the potential data loss will be zero or near zero with Data Guard.

Oracle Exadata Database Machine Inherent Benefits

Oracle Exadata Database Machine is the best Oracle Maximum Availability Architecture database platform that Oracle offers. Exadata is engineered with hardware, software, database, availability, and extreme performance for all workloads and scalability innovations that support the most mission-critical enterprise applications.

Specifically, Exadata provides unique high availability, data protection, and quality-of-service capabilities that set Oracle apart from any other platform or cloud vendor. Sizing Exadata cloud systems to meet your application and database system resource needs (for example, sufficient CPU, memory, and I/O resources) is very important to maintain the highest availability, stability, and performance. Proper sizing and resource management are especially important when consolidating many databases on the same cluster. Database consolidation is a very common benefit when leveraging Exadata.

Examples of these benefits include:

- **High availability and low brownout:** Fully redundant, fault-tolerant hardware exists in the storage, network, and database servers. Resilient, highly-available software, such as Oracle Real Application Clusters (Oracle RAC), Oracle Clusterware, Oracle Database, Oracle Automatic Storage Management (ASM), Oracle Linux, and Oracle Exadata Storage Server enables applications to maintain application service levels through unplanned outages and planned maintenance events.

  For example, Exadata has instant failure detection that can detect and repair database nodes, storage servers, and network failures in less than two seconds and resume application and database service uptime and performance. Other platforms can experience 30 seconds, or even minutes, of blackout and extended application brownouts for the same type of failures. Only the Exadata platform offers a wide range of unplanned outages and planned maintenance tests to evaluate end-to-end application and database brownouts and blackouts.

- **Data protection:** Exadata provides Oracle Database with physical and logical block corruption prevention, detection, and, in some cases, automatic remediation.

  The Exadata Hardware Assisted Resilient Data (HARD) checks include support for server parameter files, control files, log files, Oracle data files, and Oracle Data Guard broker files when those files are stored in Exadata storage. This intelligent Exadata storage validation stops corrupted data from being written to disk when a HARD check fails, which eliminates a large class of failures that the database industry had previously been unable to prevent.

  Examples of the Exadata HARD checks include:

  – Redo and block checksum

  - Correct log sequence
  - Block type validation
  - Block number validation
  - Oracle data structures, such as block magic number, block size, sequence number, and block header and tail data structures

  Exadata HARD checks are initiated from Exadata storage software (cell services) and work transparently after enabling a database `DB_BLOCK_CHECKSUM` parameter, which is enabled by default in the cloud. Exadata is the only platform that currently supports the HARD initiative.

  Furthermore, Oracle Exadata Storage Server provides non-intrusive, automatic hard disk scrub and repair. This feature periodically inspects and repairs hard disks during idle time. suppose bad sectors are detected on a hard disk. In that case, Oracle Exadata Storage Server automatically requests Oracle Automatic Storage Management (ASM) to repair the bad sectors by reading the data from another mirror copy.

  Finally, Exadata and Oracle ASM can detect corruptions as data blocks are read into the buffer cache and automatically repair data corruption with a good copy of the data block on a subsequent database write. This inherent intelligent data protection makes Exadata Database Machine and ExaDB-D the best data protection storage platform for Oracle databases.

  For comprehensive data protection, a Maximum Availability Architecture best practice is to use a standby database on a separate Exadata instance to detect, prevent, and automatically repair corruptions that cannot be addressed by Exadata alone. The standby database also minimizes downtime and data loss for disasters that result from site, cluster, and database failures.

- **Response time quality of service:** Only Exadata has end-to-end quality-of-service capabilities to ensure that response time remains low and optimum. Database server I/O latency capping and Exadata storage I/O latency capping ensure that read or write I/O can be redirected to partnered cells when response time exceeds a certain threshold. More importantly, memory and flash are intelligently pre-warmed for various maintenance events and sick component outages (“gray area outages”) to preserve application response time and performance. This end-to-end holistic performance view is a big benefit for Oracle enterprise customers who require consistent application response time and high throughput.

  Suppose storage becomes unreliable (but not failed) because of poor and unpredictable performance. In that case, the disk or flash cache can be confined offline and later returned online if heuristics show that I/O performance is back to acceptable levels. Resource management can help prioritize critical database network or I/O functionality so that your application and database perform at an optimized level.

  For example, database log writes get priority over backup requests on the Exadata network and storage. Furthermore, rapid response time is maintained during storage software updates by ensuring that the partner flash cache is warmed so flash misses are minimized.

- **End-to-end testing and holistic health checks:** Because Oracle owns the entire Oracle Exadata Cloud Infrastructure, end-to-end testing, and optimizations benefit every Exadata customer around the world, whether hosted on-premises or in the cloud. Validated optimizations and fixes required to run any mission-critical system are uniformly applied after rigorous testing. Health checks are designed to evaluate the entire stack.

  The Exadata health check utility EXACHK is Exadata cloud-aware and highlights any configuration and software alerts that may have occurred because of customer changes. No other cloud platform currently has this kind of end-to-end health check available. Oracle

recommends running EXACHK at least once a month, and before and after any software updates, to evaluate any new best practices and alerts.

- **Higher Uptime:** The uptime service-level agreement per month is 95% (a maximum of 22 minutes of downtime per month), but when you use MAA best practices for continuous service, most months would have zero downtime. With Gold MAA, you can fail over to your standby database for various disaster events such as database, cluster, or data center (or AZ) failures, depending on your standby database placement. Note setting automatic failover to your target standby with Data Guard Fast-Start Failover is a manual setup (see Configure Fast Start Failover).

Expected Impact During Unplanned Outages

The following table lists various unplanned outage events and the associated potential database downtime, application Recovery Time Objective (RTO), and data loss potential or recovery point objective (RPO).

For Oracle Data Guard architectures (MAA Gold), the database downtime or service level downtime does not include detection time or the time it takes before a customer initiates the Cloud Console Data Guard failover operation.

Outage Event	Database Downtime	Service-Level Downtime (RTO)	Potential Service-Level Data Loss (RPO)
Localized events, including: Exadata cluster network topology failures Storage (disk, flash, and storage cell) failures Database instance failures Database server failures	Zero	Near-zero	Zero
Events that require restoration from backup because a standby database does not exist: Data corruptions Full database failures Complete storage failures Availability Zone failures	Minutes to hours (without Data Guard)	Minutes to hours (without Data Guard)	30 minutes (without Data Guard)
Events using Data Guard to fail over: Data corruptions Full database failures Complete storage failures Availability Zone failures	Seconds to minutes[1] Zero downtime for physical corruptions due to the auto-block repair feature	Seconds to minutes[1] The foreground process that detects the physical corruption pauses while auto-block repair completes	Zero for Maximum Availability (SYNC redo transport) Near Zero for Maximum Performance (ASYNC redo transport)

[1]For MAA Gold, to protect your database from regional failure, instantiate the standby database in a region different from the primary database. For this MAA evaluation, the standby database was in a different AZ. Also, Data Guard Fast-Start Failover and its Data Guard observers must be set up manually to perform automatic database failover. Application workloads as high as 300 MB/second per Oracle Real Application Cluster instance were validated. The standby database was up-to-date with near-zero lag. Depending on the workload, standby database tuning may be required for extreme workloads (see Tune and Troubleshoot Oracle Data Guard).

Expected Impact During Planned Maintenance

The following tables describe the impact of various planned maintenance events for Oracle Exadata Database Service on Dedicated Infrastructure on Oracle Database@Azure.

Impact of Exadata Cloud Software Updates

The following table lists various software updates and their impact on the associated database and application. This is applicable for Oracle Exadata Database Service on Dedicated Infrastructure on Oracle Database@Azure.

Software Update	Database Impact	Application Impact	Scheduled By	Performed By
Exadata Network Fabric Switches	Zero downtime with no database restart	Zero to single-digit seconds brownout	Oracle schedules based on customer preferences, and customers can reschedule	Oracle Cloud Operation
Exadata Storage Servers	Zero downtime with no database restart	Zero to single-digit seconds brownout Exadata storage servers are updated in a rolling manner, maintaining redundancy. Oracle Exadata System Software pre-fetches the secondary mirrors of the OLTP data most frequently accessed into the flash cache, maintaining application performance during storage server restarts. Exadata smart flash for database buffers is maintained across a storage server restart. With Exadata 21.2 software, Persistent Storage Index and Persistent Columnar Cache features enable consistent query performance after a storage server software update.	Oracle schedules based on customer preferences, and customers can reschedule	Oracle Cloud Operation
Exadata Database Host Monthly Infrastructure Security Maintenance	Zero downtime with no host or database restart	Zero downtime	Oracle schedules, and customers can reschedule	Oracle Cloud Operation

Software Update	Database Impact	Application Impact	Scheduled By	Performed By
Exadata Database Host Quarterly Infrastructure Maintenance	Zero downtime with Oracle RAC rolling updates	Zero downtime Exadata Database compute resources are reduced until planned maintenance is completed.	Oracle schedules based on customer preferences, and customers can reschedule	Oracle Cloud Operation
Exadata Database Guest	Zero downtime with Oracle RAC rolling updates	Zero downtime Exadata Database compute resources are reduced until planned maintenance is completed.	Customer	Customers, using Oracle Cloud Console or APIs
Oracle Database quarterly update or custom image update	Zero downtime with Oracle RAC rolling updates	Zero downtime Exadata Database compute resources are reduced until planned maintenance is completed. Special consideration is required during rolling database quarterly updates for applications that use database OJVM (see My Oracle Support Doc ID 2217053.1 for details).	Customer	Customers using Oracle Cloud Console, APIs, or `dbaascli` utility In-place, with database home patch, and out-of-place with database move (recommended) Works for Data Guard and standby databases (see My Oracle Support Doc ID 2701789.1)
Oracle Grid Infrastructure quarterly update or upgrade	Zero downtime with Oracle RAC rolling updates	Zero downtime Exadata Database compute resources are reduced until planned maintenance is completed.	Customer	Customers, using Oracle Cloud Console, APIs, or `dbaascli` utility
Oracle Database upgrade with downtime	Minutes to Hour(s) downtime	Minutes to Hour(s) downtime	Customer	Customers, using Oracle Cloud Console, APIs, or `dbaascli` utility Works for Data Guard and standby databases (see My Oracle Support Doc ID 2628228.1)
Oracle Database upgrade with near-zero downtime	Minimal downtime with `DBMS_ROLLING`, Oracle GoldenGate replication, or with pluggable database relocate	Minimal downtime with `DBMS_ROLLING`, Oracle GoldenGate replication, or with pluggable database relocate	Customer	Customers, using `dbaascli` leveraging `DBMS_ROLLING` (see My Oracle Support Doc ID 2832235.1) Customers, using generic Maximum Availability Architecture best practices

Impact of Exadata Elastic Operations

Exadata cloud systems have many elastic capabilities that can be used to adjust database and application performance needs. By rearranging resources on need, you can maximize system resources to targeted databases and applications and minimize costs.

The following table lists elastic Oracle Exadata Cloud Infrastructure and VM Cluster updates and the impacts associated with those updates on databases and applications. All of these operations can be performed using Oracle Cloud Console or APIs unless specified otherwise.

VM Cluster Change	Database Impact	Application Impact
Scale Up or Down VM Cluster Memory	Zero downtime with Oracle RAC rolling updates	Zero to single-digit seconds brownout
Scale Up or Down VM Cluster CPU	Zero downtime with no database restart	Zero downtime Available CPU resources can impact application performance and throughput
Scale Up or Down (resize) ASM Storage for Database usage	Zero downtime with no database restart	Zero downtime Application performance might be minimally impacted
Scale Up VM Local /u02 File System Size (Exadata X9M and later systems)	Zero downtime with no database restart	Zero downtime
Scale Down VM Local /u02 File System Size	Zero downtime with Oracle RAC rolling updates for scaling down	Zero to single-digit seconds brownout
Adding Exadata Storage Cells	Zero downtime with no database restart	Zero to single-digit seconds brownout Application performance might be minimally impacted
Adding Exadata Database Servers	Zero downtime with no database restart	Zero to single-digit seconds brownout Adding Oracle RAC instances and CPU resources may improve application performance and throughput
Adding Database Nodes in Virtual Machines (VMs) Cluster	Zero downtime with no database restart	Zero to single-digit seconds brownout Application performance and throughput may increase or decrease by adding or dropping Oracle RAC instances and CPU resources

Planning for the Impact of Exadata Elastic Operations

Because some of the above elastic changes may take significant time, and they impact the available resources for your application, some planning is required.

Note that “scale down” and “drop” changes will decrease available resources. Care must be taken to not reduce resources below the amount required for database and application stability and to meet application performance targets. The following table provides you with the estimated time duration and planning recommendations for these changes.

VM Cluster Change	Database Impact	Application Impact
Scale Up or Down VM Cluster Memory	Time to drain services and Oracle RAC rolling restart Typically 15-30 minutes per node, but may vary depending on application draining	Understanding application draining See Configuring Continuous Availability for Applications before scaling down memory, ensure that database SGAs can still be stored in hugepages, and that application performance is still acceptable. To preserve predictable application performance and stability: • Monitor and scale up before important high workload patterns require the memory resources • Avoid memory scale down unless all of the database SGA and PGA memory fits into the new memory size, and the system's hugepages accommodate all SGAs
Scale Up or Down VM Cluster CPU	Online operation, typically less than 5 minutes for each VM cluster Scaling up from a very low value to a very high value (10+ OCPUs increase) may take 10 minutes.	To preserve predictable application performance and stability: • Monitor and scale up before important high workload patterns require the CPU resources, or when consistently reaching an OCPU threshold for a tolerated amount of time • Only scale down if the load average is below the threshold for at least 30 minutes, or scale down based on fixed workload schedules (such as business hours with 60 OCPUs, non-business hours with 10 OCPUs, and batch with 100 OCPUs) • Avoid more than one scale-down request within a 2 hour period
Scale Up or Down (resize) ASM Storage for Database usage	Typically minutes to hours Time varies based on utilized database storage capacity and database activity. The higher the percentage of utilized database storage, the longer the resize operation (which includes ASM rebalance) will take.	Oracle ASM rebalance is initiated automatically. Storage redundancy is retained. Because of the inherent best practice of using a non-intrusive ASM power limit, application workload impact is minimal. Choose a non-peak window so resize and rebalance operations can be optimized. Because the time may vary significantly, plan for the operation to be completed in hours. To estimate the time that an existing resize or rebalance operation per VM cluster requires, query `GV$ASM_OPERATION`. For example, you can run the following query every 30 minutes to evaluate how much work (`EST_WORK`) and how much more time (`EST_MINUTES`) potentially is required: `select operation, pass, state, sofar, est_work, est_minutes from gv$asm_operation where operation='REBAL';` Note that the estimated statistics tend to become more accurate as the rebalance progresses, but can vary based on the concurrent workload.

VM Cluster Change	Database Impact	Application Impact
Scale Up VM Local /u02 File System Size (Exadata X9M and later)	Online operation, typically less than 5 minutes for each VM cluster	VM local file system space is allocated on local database host disks, which are shared by all VM guests for all VM clusters provisioned on that database host. Do not scale up space for Local /u02 File System unnecessarily on one VM cluster, such that no space remains to scale up on other VM clusters on the same Exadata Infrastructure, because a Local /u02 File System scale down must be performed in an Oracle RAC rolling manner, which may cause application disruption.
Scale Down VM Local /u02 File System Size	Time to drain services and Oracle RAC rolling restart Typically 15-30 minutes for each node, but may vary depending on application draining settings.	To plan, learn about application draining at Configuring Continuous Availability for Applications
Adding Exadata Storage Cells	The online operation creates more available space for administrators to choose how to distribute. Typically, 3-72 hours per operation, depending on the number of VM clusters, database storage usage, and storage activity. With a very active database and heavy storage activity, this can take up to 72 hours. As part of the add storage cell operation, there are two parts to this operation: 1. Storage is added to the Exadata system as part of the add storage operation. 2. The administrator must decide which VM cluster to expand its ASM disk groups as a separate operation.	Plan to add storage when your storage capacity utilization hits 80% within a month, because this operation may be completed in days. Oracle ASM rebalance is initiated automatically. Storage redundancy is retained. Because of inherent best practices in using non-intrusive ASM power limits, the impact of application workload is minimal. Because the time duration may vary significantly, plan for the operation to be completed in days before the storage is available. To estimate the time that an existing resize or rebalance operation will take on each VM cluster, query `GV$ASM_OPERATION`. For example, you can run the following query every 30 minutes to evaluate how much work (`EST_WORK`) and how much more time (`EST_MINUTES`) is potentially required: `Select operation, pass, state, sofar, est_work, est_minutes from gv$asm_operation where operation='REBAL';` Note that the estimated statistics tend to become more accurate as the rebalance progresses, but can vary based on the concurrent workload.
Adding Exadata Database Servers	Online operation to expand your VM cluster One-step process to add the Database Compute to the Exadata infrastructure and then expand the VM cluster Approximately 1 to 6 hours for each Exadata Database Server	Plan to add Database Compute when your database resource utilization reaches 80% within a month. Be aware, and plan for this operation to take many hours to a day. Choose a non-peak window so that the add Database Compute operation can be completed faster. Each Oracle RAC database registered by Oracle Clusterware and visible in the Oracle Cloud Console is extended. If a database was configured outside the Oracle Cloud Console, or without dbaascli, then those databases will not be extended.

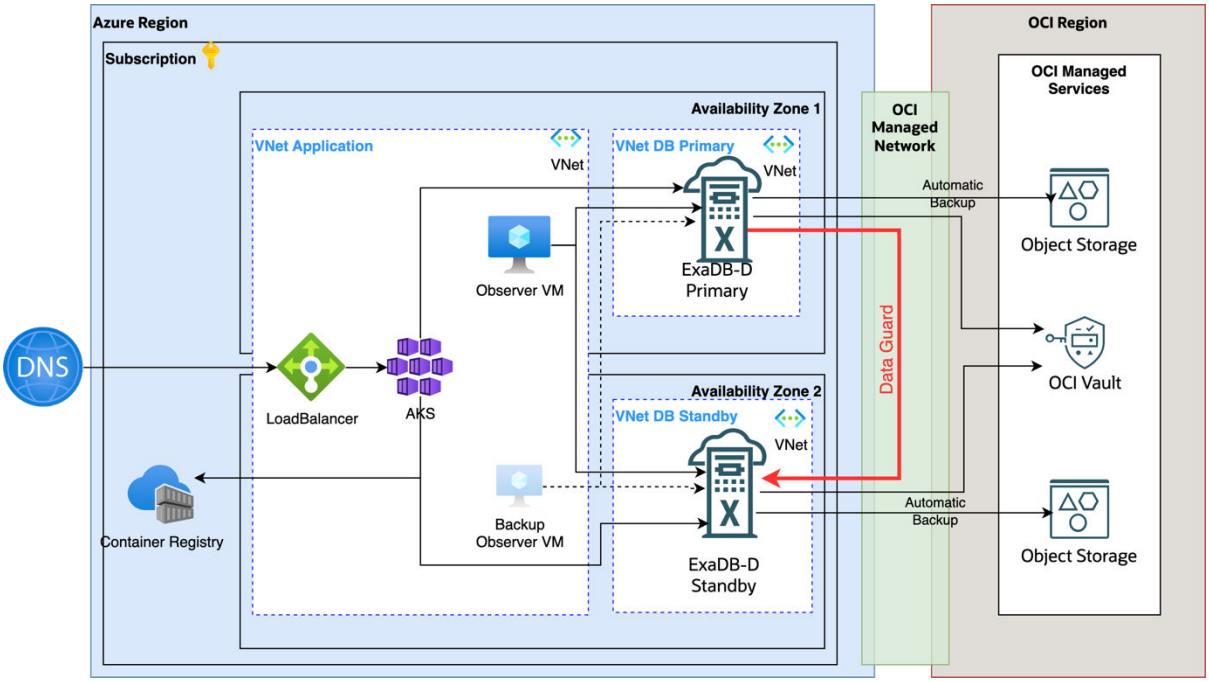
VM Cluster Change	Database Impact	Application Impact
Adding or Dropping Database Nodes in Virtual Machines (VMs) Cluster	Zero database downtime when adding Database Nodes in the VM cluster. Typically takes 3-6 hours, depending on the number of databases in the VM cluster. Zero database downtime when dropping Database Nodes in the VM cluster. Typically takes 1-2 hours, depending on the number of databases in the VM cluster.	Understand that the add/drop operation is not instantaneous, and the operation may take several hours to complete. The drop operation reduces database computing, OCPU, and memory resources so that application performance can be impacted.

MAA Gold Network Topology and Evaluation

The recommended MAA Gold architecture on Oracle Database@Azure consists of:

- When using Data Guard, Oracle Exadata infrastructures (ExaDB-D) are provisioned in two different Availability Zones (AZ) using separate VNets that do not have overlapping IP CIDR ranges.
- Backup network subnets assigned to the primary and standby clusters do not have overlapping IP CIDR ranges.
- The application tier spans at least two AZs, and the VNet is peered with each VNet of primary and standby VM Clusters.
- Database backups and restore operations use a high bandwidth network for OCI Object Storage.

Figure 31-1 DR Capability for a Solution in the Same Region



Application Network Layer on Azure

The proximity of the application tier to the database cluster affects application response time.

If you require a very low latency response time (for example, 200-400 microseconds), deploy the application VMs in the same AZ as the database cluster. Latency increases to possibly 1 millisecond or more when application and database servers are configured across VNets or AZs.

Deploy the application tier over at least two AZs for High Availability. The deployment process and solution over multiple AZs vary depending on the application's components, Azure services, and resources involved. For example, with Azure Kubernetes Services (AKS), you can deploy the worker nodes in different AZs. Kubernetes control plane maintains and synchronizes the pods and the workload.

Database Network Layer

Oracle Data Guard maintains a standby database by transmitting and applying redo data from the primary database. Use Data Guard switchover for planned maintenance or disaster recovery tests. If the primary database becomes unavailable, use Data Guard failover to resume service.

Peering Networks Between Primary and Standby

The primary and standby Exadata Clusters are deployed in separate networks. Oracle Database@Azure Exadata Clusters are always deployed using separate Virtual Cloud Networks (VCN) in OCI. These separate VCNs must be connected to allow traffic to pass between them, that is they must be "peered" before enabling Data Guard with Oracle cloud automation. For this reason, the networks must use separate, non-overlapping IP CIDR ranges.

Peering can be done using the OCI network or Azure network. The recommended option is to peer the OCI VCNs and use the OCI network for redo traffic. OCI VCN peering provides higher single-process network throughput (observed up to 14 Gbits/s), lower latency between database clusters, and there is no chargeback for this traffic. Peering using the Azure network provides an observed 3 Gbit/s single process throughput (relevant for database instances with high redo generation rates over 300 MB/s), has approximately 20% higher latency, and there is a chargeback for cross-VNet traffic.

Recommended OCI VCN Peering for Data Guard

When Exadata Clusters are created in Azure, each cluster is in a different Virtual Cloud Network (VCN) in OCI. Connectivity between VCNs is required for Data Guard redo transport. This connectivity, or peering, must be configured before enabling Data Guard in Oracle Database@Azure. For resources in different VCNs to communicate with each other, as is required by Data Guard, additional steps are required to peer the VCNs and allow the IP address ranges access to each other.

Follow these high-level steps to peer the VCNs. More details are available at Configure VCN peering (oracle.com).

1. Provision a Local Peering Gateway in each VCN.
2. Establish a peer connection between Local Peering Gateways.
3. Update the default route table to route traffic between VCNs.
4. Update VCN Network Security Groups (NSG) to allow connections.

Alternative Option of Azure VNet Peering for Data Guard

To peer the Azure VNets for Data Guard redo traffic, see https://learn.microsoft.com/en-us/azure/virtual-network/virtual-network-peering-overview.

Be aware that when networks are peered through Azure, latency increases by about 20%, and single-process network throughput is limited to approximately 3 Gbit/s (~375 MB/sec). This is relevant because Data Guard redo transport is a single process for each database instance; therefore, if a single instance produces redo at a higher rate, a transport lag may form. There is an additional cost for ingress and egress network traffic in each VNet when networks are peered through Azure.

Enable Data Guard

After the network is peered by one of the above options, you can Enable Data Guard (see Use Oracle Data Guard with Exadata Cloud Infrastructure).

Network Throughput and Latency Evaluation

When comparing throughput and latency between networks, the following methods are recommended.

Data Guard Throughput

It is recommended that iperf be used to measure throughput between endpoints.

Examples:

Server side (as root):

```
# iperf -s
```

Client Side (as root):

Single process: `iperf -c <ip address of VIP>`

- This determines the maximum redo throughput from one Oracle RAC instance to a standby Oracle RAC instance.
- single-process network throughput estimated to be 14 Gbits/s with OCI VCN Peering
- single-process network throughput estimated to be 3 Gbits/s with Azure VNet Peering

Parallel process: `iperf -c <ip address of VIP> -P 32`

- This determines the maximum network bandwidth available for Data Guard instantiation and large redo gap resolution.

Backups

For backups, RMAN nettest was used and met the expected results. See My Oracle Support Doc ID 2371860.1 for details about nettest.

Oracle database backup and restore throughput to Oracle's Object Storage Service were within performance expectations. For example, an ExaDB-D 2 node cluster (using 16+ OCPUs) and 3 storage cells may observe a 4 TB/hour backup rate and approximately 8 TB/hour restore rate with no other workloads running on the cluster. By increasing the RMAN channels, you can leverage available network bandwidth or storage bandwidth and achieve as much as 42 TB/hour backup rate and 8.7 TB/hour restore rate. The performance varies based on existing workloads and network traffic on the shared infrastructure.

Latency

The best tool for testing TCP latency between VM endpoints is sockperf. Latency is not tested for backups. sockperf is not installed by default and must be installed from an RPM or YUM.

server: `sockperf sr -i <IP of VIP> --tcp`

client: `sockperf pp -i <IP of VIP> --tcp --full-rtt`

Sample output(client) between clusters in different AZs:

```
# sockperf pp -i <IP> --tcp --full-rtt
  sockperf: Summary: Round trip is 1067.225 usec
  sockperf: Total 516 observations; each percentile contains 5.16 observations
  sockperf: ---> <MAX> observation = 1194.612
  sockperf: ---> percentile 99.999 = 1194.612
  sockperf: ---> percentile 99.990 = 1194.612
  sockperf: ---> percentile 99.900 = 1137.864
  sockperf: ---> percentile 99.000 = 1112.276
  sockperf: ---> percentile 90.000 = 1082.640
  sockperf: ---> percentile 75.000 = 1070.377
  sockperf: ---> percentile 50.000 = 1064.075
  sockperf: ---> percentile 25.000 = 1059.195
  sockperf: ---> <MIN> observation = 1047.373
```

> **Note:**
>
> Results vary based on region and AZ sampled.
>
> The ping command should not be used in Azure because ICMP packets are set to very low priority and will not accurately represent the latency of TCP packets.

Traceroute

Run traceroute between endpoints to ensure that the proper route is being taken.

Observations

- One ‘hop’ between ExaDB-D clusters when Data Guard uses OCI VCN peering
- Six ‘hops’ between ExaDB-D clusters when Data Guard uses Azure VNet peering
- Four ‘hops’ between application VMs and ExaDB-D clusters in the same AZ

Achieving Continuous Availability For Your Applications

As part of Oracle Exadata Database Service on Dedicated Infrastructure on Oracle Database@Azure, all software updates (except for non-rolling database upgrades or non-rolling patches) can be done online or with Oracle RAC rolling updates to achieve continuous database uptime.

Furthermore, any local failures of storage, Exadata network, or Exadata database server are managed automatically, and database uptime is maintained.

To achieve continuous application uptime during Oracle RAC switchover or failover events, follow these application-configuration best practices:

- Use Oracle Clusterware-managed database services to connect your application. For Oracle Data Guard environments, use role-based services.
- Use the recommended connection string with built-in timeouts, retries, and delays so that incoming connections do not see errors during outages.
- Configure your connections with Fast Application Notification.
- Drain and relocate services. Use the recommended best practices in the table below that support draining, such as test connections, when borrowing or starting batches of work, and return connections to pools between uses.
- Leverage Application Continuity or Transparent Application Continuity to replay in-flight uncommitted transactions transparently after failures.

For more details, see Configuring Continuous Availability for Applications. Oracle recommends testing your application readiness by following Validating Application Failover Readiness (My Oracle Support Doc ID 2758734.1).

Depending on the Oracle Exadata Database Service planned maintenance event, Oracle attempts to automatically drain and relocate database services before stopping any Oracle RAC instance. For OLTP applications, draining and relocating services typically work very well and result in zero application downtime.

Some applications, such as long-running batch jobs or reports, may not be able to drain and relocate gracefully within the maximum draining time. For those applications, Oracle recommends scheduling the software planned maintenance window around these types of activities or stopping these activities before the planned maintenance window. For example, you can reschedule a planned maintenance window to run outside your batch windows or stop batch jobs before a planned maintenance window.

Special consideration is required during rolling database quarterly updates for applications that use database OJVM. See My Oracle Support Doc ID 2217053.1 for details.

The following table lists planned maintenance events that perform Oracle RAC instance rolling restart, as well as the relevant service drain timeout variables that may impact your application.

Exadata Cloud Software Updates or Elastic Operation	Drain Timeout Variables
Oracle DBHOME patch apply and database MOVE	Cloud software automation stops/relocates database services while honoring `DRAIN_TIMEOUT` settings defined by database service configuration (such as `srvctl`).[1] You can override `DRAIN_TIMEOUT` defined on services using the option `drainTimeoutInSeconds` with command line operation `dbaascli dbHome patch` or `dbaascli database move`. The Oracle Cloud internal maximum draining time supported is 2 hours.

Exadata Cloud Software Updates or Elastic Operation	Drain Timeout Variables
Oracle Grid Infrastructure (GI) patch apply and upgrade	Cloud software automation stops/relocates database services while honoring `DRAIN_TIMEOUT` settings defined by database service configuration (such as `srvctl`).[1] You can override the `DRAIN_TIMEOUT` defined on services using the option `drainTimeoutInSeconds` with command line operation `dbaascli grid patch` or `dbaascli grid upgrade`. The Oracle Cloud internal maximum draining time supported is 2 hours.
Virtual Machine Operating System Software Update (Exadata Database Guest)	Exadata `patchmgr/dbnodeupdate` software program calls drain orchestration (`rhphelper`). Drain orchestration has the following drain timeout settings (see My Oracle Support Doc ID 2385790.1 for details): • `DRAIN_TIMEOUT` – If a service does not have `DRAIN_TIMEOUT` defined, then the default value of 180 seconds is used. • `MAX_DRAIN_TIMEOUT` - Overrides any higher `DRAIN_TIMEOUT` value defined by database service configuration. The default value is 300 seconds. There is no maximum value. The `DRAIN_TIMEOUT` settings defined by the database service configuration are honored during service stop/relocate.
Exadata X9M and later systems Scale Down VM Local File System Size	Exadata X9M and later systems call drain orchestration (`rhphelper`). Drain orchestration has the following drain timeout settings (see My Oracle Support Doc ID 2385790.1 for details): • `DRAIN_TIMEOUT` – If a service does not have `DRAIN_TIMEOUT` defined, then the default value of 180 seconds is used. • `MAX_DRAIN_TIMEOUT` - Overrides any higher `DRAIN_TIMEOUT` value defined by database service configuration. The default value is 300 seconds. The `DRAIN_TIMEOUT` settings defined by the database service configuration are honored during service stop/relocate. The Oracle Cloud internal maximum draining time supported for this operation is 300 seconds.

Exadata Cloud Software Updates or Elastic Operation	Drain Timeout Variables
Exadata X9M and later systems Scale Up or Down VM Cluster Memory	Exadata X9M and later systems call drain orchestration (`rhphelper`). Drain orchestration has the following drain timeout settings (see My Oracle Support Doc ID 2385790.1 for details): • `DRAIN_TIMEOUT` – If a service does not have `DRAIN_TIMEOUT` defined, then the default value of 180 seconds is used. • `MAX_DRAIN_TIMEOUT` - Overrides any higher `DRAIN_TIMEOUT` value defined by database service configuration. The default value is 300 seconds. The `DRAIN_TIMEOUT` settings defined by the database service configuration are honored during service stop/relocate. The Oracle Cloud internal maximum draining time supported for this operation is 900 seconds.
Oracle Exadata Cloud Infrastructure (ExaDB) software update	The ExaDB-D database host calls drain orchestration (`rhphelper`). Drain orchestration has the following drain timeout settings (see My Oracle Support Doc ID 2385790.1 for details): • `DRAIN_TIMEOUT` – If a service does not have `DRAIN_TIMEOUT` defined, then the default value of 180 seconds is used. • `MAX_DRAIN_TIMEOUT` - Overrides any higher `DRAIN_TIMEOUT` value defined by database service configuration. The default value is 300 seconds. The `DRAIN_TIMEOUT` settings defined by the database service configuration are honored during service stop/relocate. The Oracle Cloud internal maximum draining time supported for this operation is 500 seconds. Enhanced Infrastructure Maintenance Controls: To achieve draining time longer than the Oracle Cloud internal maximum, leverage the custom action capability of Enhanced Infrastructure Maintenance Controls, which allows you to suspend infrastructure maintenance before the next database server update starts, directly stop/relocate database services running on the database server, and then resume infrastructure maintenance to proceed to the next database server. See Configure Oracle-Managed Infrastructure Maintenance in Oracle Cloud Infrastructure documentation for details.

[1]Minimum software requirements to achieve this service drain capability are: Oracle Database release 12.2 and later and the latest cloud DBaaS tooling software.

Oracle MAA Reference Architectures in Oracle Exadata Cloud

Oracle Exadata Database Service on Dedicated Infrastructure on Oracle Database@Azure supports all Oracle MAA reference architectures, providing support for all Oracle databases, regardless of their specific high availability, data protection, and disaster recovery service-level agreements.

See MAA Best Practices for the Oracle Cloud for more information about Oracle MAA in the Oracle Exadata Cloud.

32

Oracle Data Guard Hybrid Cloud Configuration

A hybrid Oracle Data Guard configuration consists of a primary database and one or more standby databases residing partially on-premises and partially in the cloud. The process detailed here uses the Oracle Zero Downtime Migration tool to create a cloud standby database from an existing on-premises primary database.

Zero Downtime Migration streamlines and simplifies the process of creating the standby database on the cloud, while incorporating MAA best practices

After establishing the cloud standby database as described here, you can perform a role transition so that the primary database runs in the cloud instead of on-premises.

Benefits Of Hybrid Data Guard in the Oracle Cloud

The following are the primary benefits to using a hybrid Data Guard configuration in the Oracle Cloud.

- Oracle manages the cloud data center and infrastructure.
- Ability to switch over (planned events) or fail over (unplanned events) production to the standby database in the cloud during scheduled maintenance or unplanned outages. Once a failed on-premises database is repaired, it can be synchronized with the current production database in the cloud. Then, production can be switched back to the on-premises database.
- Use the same Oracle MAA best practices as the on-premises deployment. Additional Oracle MAA best practices specific to hybrid Data Guard deployments are specified in the topics that follow. When configured with MAA practices, a hybrid Data Guard configuration provides:
  - Recovery Time Objective (RTO) of seconds with automatic failover when configured with Data Guard fast start failover
  - Recovery Point Objective (RPO) less than a second for Data Guard with ASYNC transport
  - RPO zero for Data Guard in a SYNC or FAR SYNC configuration

> **Note:**
>
> Data Guard life cycle management operations, such as switchover, failover, and reinstatement, are manual processes in a hybrid Data Guard configuration.

MAA Recommendations for using Exadata Cloud for Disaster Recovery

When deploying Exadata Cloud for Disaster Recovery, Oracle MAA recommends:

- Create a cloud database system target that is symmetric or similar to the on-premises primary database to ensure performance SLAs can be met after a role transition. For example, create an Oracle RAC target for an Oracle RAC source, Exadata for Exadata, and so on.
- Ensure that network bandwidth can handle peak redo rates in addition to existing network traffic.

  My Oracle Support document Assessing and Tuning Network Performance for Data Guard and RMAN (Doc ID 2064368.1) provides additional network bandwidth troubleshooting guidance for assessing and tuning network performance for Data Guard and RMAN.
- Ensure network reliability and security between on-premises and the Cloud environment.
- Use Oracle Active Data Guard for additional automatic block repair, data protection, and offloading benefits.
- Use Oracle Transparent Data Encryption (TDE) for both primary and standby databases.

  My Oracle Support document Oracle Database Tablespace Encryption Behavior in Oracle Cloud (Doc ID 2359020.1) has additional details on TDE behavior in cloud configurations.
- Configure backups to object storage or Autonomous Recovery Service for the database in OCI or Azure, in primary or standby role. See Manage Database Backup and Recovery on Oracle Exadata Database Service on Dedicated Infrastructure and Database Autonomous Recovery Service.

Service Level Requirements

Oracle Data Guard hybrid deployments are user-managed environments. The service level expectations for availability, data protection, and performance that are practical for a given configuration and application must be determined by your requirements.

Service levels must be established for each of the following dimensions relevant to disaster recovery that are applicable to any Data Guard configuration:

- **Recovery Time Objective (RTO)** describes the maximum acceptable downtime if an outage occurs. This includes the time required to detect the outage and to fail over the database and application connections so that service is resumed.
- **Recovery Point Objective (RPO)** describes the maximum amount of data loss that can be tolerated. Achieving the desired RPO depends on:
  - Available bandwidth relative to network volume
  - The ability of the network to provide reliable, uninterrupted transmission
  - The Data Guard transport method used: asynchronous for near-zero data loss protection, synchronous for zero data loss protection
- **Data Protection** - You can configure the most comprehensive block corruption detection, prevention, and auto-repair with Oracle Active Data Guard and MAA.
- **Performance** - Database response time may be different after a fail over if not enough capacity for compute, memory, I/O, and so on, is provisioned at the standby system, compared to the on-premises production system.

  This occurs when administrators intentionally under-configure standby resources to reduce cost, accepting a reduced service level while in DR mode. MAA best practices recommend configuring symmetrical capacity on both the primary and standby database hosts so there is no change in response time after a fail over.

Rapid provisioning available with the cloud facilitates a middle ground where there is less capacity deployed during steady-state, but the new primary database system is rapidly scaled-up should a fail over be required.

> **Note:**
>
> The reduced resources during steady state in a rapid provisioning approach could impact the ability of recovery to keep the standby database current with the primary database, creating an apply lag and impacting RTO. This approach should only be considered after thorough testing.

See High Availability and Data Protection – Getting From Requirements to Architecture for more details about determining RTO and RPO requirements along with other considerations.

See Detecting and Monitoring Data Corruption .

Security Requirements and Considerations

Oracle MAA best practices recommend using Oracle Transparent Data Encryption (TDE) to encrypt the primary and standby databases to ensure that data is encrypted at-rest.

Using TDE to protect data is an essential part of improving the security of the system; however, you must be aware of certain considerations when using any encryption solution, including:

- **Additional CPU overhead** - Encryption requires additional CPU cycles to calculate encrypted and decrypted values. TDE, however, is optimized to minimize the overhead by taking advantage of database caching capabilities and leveraging hardware acceleration within Exadata. Most TDE users see little performance impact on their production systems after enabling TDE.
- **Lower data compression** - Encrypted data compresses poorly because it must reveal no information about the original plain text data, so any compression applied to data encrypted with TDE has low compression ratios.

  When TDE encryption is used, redo transport compression is not recommended; however, when TDE is used in conjunction with Oracle Database compression technologies such as Advanced Compression or Hybrid Columnar Compression, compression is performed before the encryption occurs, and the benefits of compression and encryption are both achieved.
- **Key management** - Encryption is only as strong as the encryption key used and the loss of the encryption key is tantamount to losing all data protected by that key.

  If encryption is enabled on a few databases, keeping track of the key and its life cycle is relatively easy. As the number of encrypted databases grows, managing keys becomes an increasingly difficult problem. If you are managing a large number of encrypted databases, it is recommended that Oracle Key Vault be used on-premises to store and manage TDE master keys.

Data can be converted during the migration process, but it is recommended that TDE be enabled before beginning the migration to provide the most secure Oracle Data Guard environment. A VPN connection or Oracle Net encryption is also required for inflight encryption for any other database payload that is not encrypted by TDE, such as data file or redo headers for example. See My Oracle Support document Oracle Database Tablespace Encryption Behavior in Oracle Cloud (Doc ID 2359020.1) for more information.

If the on-premises database is not already enabled with TDE, see My Oracle Support document Primary Note For Transparent Data Encryption (TDE) (Doc ID 1228046.1) to enable TDE and create wallet files.

If TDE cannot be enabled for the on-premises database, see Encryption of Tablespaces in an Oracle Data Guard Environment in *Oracle Database Advanced Security Guide* for information about decrypting redo operations in hybrid cloud disaster recovery configurations where the Cloud database is encrypted with TDE and the on-premises database is not.

Platform, Database, and Network Prerequisites

The following requirements must be met to ensure a successful migration to a Cloud standby database.

Requirement Type	On-Premises Requirements	Oracle Cloud Requirements
Operating System	Linux, Windows or Solaris X86 (My Oracle Support Note 413484.1 for Data Guard cross-platform compatibility)	Oracle Enterprise Linux (64-bit)
Oracle Database Version*	All Oracle releases supported by Zero Downtime Migration* See Supported Database Versions for Migration for details about Oracle releases and edition support for the on-premises source database.	Extreme performance / BYOL* See Supported Database Editions and Versions for information about database service options in Oracle Cloud.
Oracle Database Architecture	Oracle RAC or single-instance	Oracle RAC or single-instance
Oracle Multitenant	For Oracle 12.1 and above, the primary database must be a multitenant container database (CDB)	Multitenant container database (CDB) or non-CDB
Physical or Virtual Host	Physical or Virtual	Exadata Virtual
Database Size	Any size	Any size. For shape limits please consult Exadata Cloud documentation
TDE Encryption	Recommended	Mandatory for Cloud databases

* The Oracle Database release on the primary and standby databases should be the same database major release and database release update (RU) during initial standby instantiation. For database software updates that are standby-first compatible, the primary and standby database Oracle Home software can be different (for example, 19RU vs 19 RU+1). For the standby instantiation in the Oracle cloud, the standby database Oracle Home software must be the same or a later RU. See Oracle Patch Assurance - Data Guard Standby-First Patch Apply (Doc ID 1265700.1).

Cloud Network Prerequisites

Data transfers from on-premises to Oracle Cloud Infrastructure (OCI) use the public network, VPN, and/or the high bandwidth option provided by Oracle FastConnect.

In an Oracle Data Guard configuration, the primary and standby databases must be able to communicate bi-directionally. This requires additional network configuration to allow access to ports between the systems.

> **Note:**
>
> Network connectivity configuration is not required for Oracle Exadata Database Service on Cloud@Customer because it is deployed on the on-premises network. Skip to On-Premises Prerequisites if using ExaDB-C@C.

Secure Connectivity

For Oracle Exadata Database Service (not required for ExaDB-C@C) there are two options to privately connect the virtual cloud network to the on-premises network: FastConnect and IPSec VPN. Both methods require a Dynamic Routing Gateway (DRG) to connect to the private Virtual Cloud Network (VCN).

See Access to Your On-Premises Network for details about creating a DRG.

- **OCI FastConnect** - Provides an easy way to create a dedicated, private connection between the data center and OCI. FastConnect provides higher bandwidth options and a more reliable and consistent networking experience compared to internet-based connections. See FastConnect Overview. (link https://docs.oracle.com/en-us/iaas/Content/Network/Concepts/fastconnectoverview.htm) for details.
- **IPSec VPN** - Internet Protocol Security or IP Security (IPSec) is a protocol suite that encrypts the entire IP traffic before the packets are transferred from the source to the destination. See Site-to-Site VPN Overview for an overview of IPSec in OCI.

Public Internet Connectivity

Connectivity between OCI and on-premises can also be achieved using the public internet.

This method is not secure by default; additional steps must be taken to secure transmissions. The steps for hybrid Data Guard configuration assume public internet connectivity.

By default, cloud security for port 1521 is disabled. Also, this default pre-configured port in the cloud for either a Virtual Machine (VM) or Bare Metal (BM) has open access from the public internet.

1. If a Virtual Cloud Network (VCN) for the standby database doesn't have an Internet Gateway, one must be added.

   To create an internet gateway see Internet Gateway.

2. Ingress and egress rules must be configured in the VCN security list to connect from and to the on-premises database.

   See Security Lists for additional information.

On-Premises Prerequisites

The following prerequisites must be met before instantiating the standby database.

Evaluate Network Using oratcptest

In an Oracle Data Guard configuration, the primary and standby databases transmit information in both directions. This requires basic configuration, network tuning, and opening of ports at both the primary and standby databases.

It is vital that the bandwidth exists to support the redo generation rate of the primary database.

Follow instructions in Assessing and Tuning Network Performance for Data Guard and RMAN (Doc ID 2064368.1) to assess and tune the network link between the on-premises and cloud environments.

Configuration

- Name resolution
  - For ExaDB-C@C, because the clusters reside on the on-premises network, the on-premises DNS should resolve each cluster, and no further configuration should be necessary.
  - For Oracle Exadata Database Service, name resolution between the clusters must be configured.

    This can be done either using a static file like /etc/hosts, or by configuring the on-premises DNS to properly resolve the public IP address of the OCI instance. In addition, the on-premises firewall must have Access Control Lists configured to allow SSH and Oracle Net to be accessed from the on-premises system to OCI.
- Oracle Data Guard in a DR configuration requires access from the Cloud instance to the on-premises database; the primary database listener port must be opened with restricted access from the Cloud IP addresses using features like iptables.

  Because every corporation has different network security policies, the network administrator must perform operations like the cloud-side network configuration shown in Cloud Network Prerequisites.
- Prompt-less SSH from Oracle Cloud to the on-premises machine. This is configured both for on-premises to Cloud during the provisioning process and from the Cloud to on-premises.
- The configuration of the on-premises firewall to allow inbound SSH connectivity from the Cloud to the on-premises machine.
- It is strongly recommended that you complete the network assessment described above in Evaluate Network Using oratcptest. Setting the appropriate TCP socket buffers sizes is especially important for ASYNC redo transport.
- It is recommended that the RDBMS software be the same on the primary and standby database for instantiation. If the current on-premises Oracle Database release is not available in Oracle Exadata Database Service, then the primary database must be on the same major database release and the same or lower Release Update (RU).

Implement MAA Best Practice Parameter Settings on the Primary Database

Most MAA best practices for Data Guard are part of the process described here; however, the Standby Redo Log should be created on the primary database before starting this process.

See Oracle Data Guard Configuration Best Practices for information.

Validating Connectivity between On-Premises and Exadata Cloud Hosts

After the networking steps are implemented successfully, run the command below to validate that the connection is successful between all sources and all targets in both directions.

On the on-premises host run:

```
[root@onpremise1 ~]# telnet TARGET-HOST-IP-ADDRESS PORT
Trying xxx.xxx.xxx.xxx...
Connected to xxx.xxx.xxx.xxx.
Escape character is '^]'.
^C^]q
```

```
telnet> q
Connection closed.
```

On the Cloud hosts run:

```
[root@oci2 ~]# telnet TARGET-HOST-IP-ADDRESS PORT
Trying xxx.xxx.xxx.xxx...
Connected to xxx.xxx.xxx.xxx.
Escape character is '^]'.
^]q
telnet> q
Connection closed.
```

If telnet is successful, proceed to the next step.

> **Note:**
>
> netcat (`nc -zv`) can be used in place of telnet.

Instantiate the Standby Using Zero Downtime Migration

Prepare the Zero Downtime Migration environment and instantiate the standby database using the physical migration method.

Each task references procedures from the latest Zero Downtime Migration documentation in *Move to Oracle Cloud Using Zero Downtime Migration* and then includes additional information pertaining to hybrid Data Guard configuration.

For the Oracle Data Guard hybrid use case, a Zero Downtime Migration migration can also be called a standby database instantiation.

After the standby database is instantiated, but before completing the full migration work flow, the migration job is stopped leaving the standby in place on the cloud. Some additional 'fix-ups' are needed to complete the hybrid Data Guard configuration.

Task 1: Install and Configure Zero Downtime Migration

The Zero Downtime Migration architecture includes a Zero Downtime Migration service host, which is separate from the primary and standby database hosts. Zero Downtime Migration software is installed and configured on the Zero Downtime Migration service host.

Any Linux Server, for example a DBCS compute resource, can be used as the service host if it meets the requirements and can be accessed bidirectionally by the target and source database systems.

See Setting Up Zero Downtime Migration Software for the host configuration and installation instructions.

Task 2: Prepare for a Physical Database Instantiation

The hybrid Data Guard configuration process uses the Zero Downtime Migration physical database online migration work flow with the option to pause the migration job after the target database instantiation.

When the standby database is instantiated and verified, the migration job can be stopped, leaving the standby database in place.

To prepare for a physical migration follow the instructions in Preparing for a Physical Database Migration in *Move to Oracle Cloud Using Zero Downtime Migration*.

Additional information specific to hybrid Data Guard configuration is detailed below.

Configuring Transparent Data Encryption on the Source Database

Transparent Data Encryption (TDE) is required on Oracle Cloud databases, including any standby database which is part of a hybrid Data Guard configuration.

While it is strongly recommended that the on-premises database also be encrypted, leaving the primary database unencrypted as part of a hybrid Data Guard configuration can be configured, and is better supported by new parameters in Oracle Database 19c (19.16) and later releases.

For all TDE configurations with Oracle Data Guard, the encryption wallet must be created on the primary database and the master key must be set.

The parameters required for TDE configuration differ depending with Oracle Database releases. The values may be different for each database in the Data Guard configuration.

- In Oracle Database release 19c (19.16) and later, the parameters `TABLESPACE_ENCRYPTION`, `WALLET_ROOT`, and `TDE_CONFIGURATION` are required to properly configure TDE.
- For Oracle Database 19c releases before 19.16, set parameters `WALLET_ROOT`, `TDE_CONFIGURATION`, and `ENCRYPT_NEW_TABLESPACES`.
- For releases earlier than Oracle Database19c, set parameters `ENCRYPTION_WALLET_LOCATION` and `ENCRYPT_NEW_TABLESPACES`.

> **Note:**
>
> Unless otherwise specified by the `TABLESPACE_ENCRYPTION=DECRYPT_ONLY` parameter, a new tablespace's encryption on the standby database will be the same as that of the primary.

In the following table use the links to find references for setting the primary and standby database parameters.

Parameters	Definition	All Oracle Database releases before 19c	Oracle Database release 19.15 and earlier	Oracle Database release 19.16 and later
ENCRYPTION_WALLET_LOCATION	Defines the location of the wallet See About the Keystore Location in the sqlnet.ora File	RECOMMENDED	DEPRECATED	DEPRECATED
WALLET_ROOT and TDE_CONFIGURATION	WALLET_ROOT sets the location of the root of the directory for wallet storage for each PDB in a CDB. See WALLET_ROOT TDE_CONFIGURATION defines the type of keystore. For example, FILE for a wallet keystore. The keystore type must be set to the same value on the primary and standby database. See TDE_CONFIGURATION	N/A	RECOMMENDED	RECOMMENDED

Parameters	Definition	All Oracle Database releases before 19c	Oracle Database release 19.15 and earlier	Oracle Database release 19.16 and later
ENCRYPT_NEW_TABLESPACES	Indicates whether a new tablespace on the primary database should be encrypted The ENCRYPT_NEW_TABLESPACES parameter can be set as follows: • CLOUD_ONLY - Default setting. Any new tablespaces created are transparently encrypted with the AES128 algorithm, unless a different algorithm is specified in the ENCRYPTION clause in the CREATE TABLESPACE statement. For on-premises databases, tablespaces are only encrypted if the CREATE TABLESPACE...ENCRYPTION clause is specified. • ALWAYS - Any new tablespace created in a primary database, on-premises or in the cloud, will be transparently encrypted with the AES128 algorithm, unless a different encryption algorithm is specified in the CREATE TABLESPACE ENCRYPTION clause. • DDL - Allows you to create tablespaces with or without encryption following the CREATE	RECOMMENDED	RECOMMENDED	NOT RECOMMENDED Override with recommended setting for TABLESPACE_ENCRYPTION

Parameters	Definition	All Oracle Database releases before 19c	Oracle Database release 19.15 and earlier	Oracle Database release 19.16 and later
	TABLESPACE statement, and also lets you change the encryption algorithm. **Note:** This value is not applicable for cloud primary databases with releases from Oracle Database 19c (19.16) and later because tablespace encryption is enforced. See ENCRYPT_NEW_TABLESPACES			
TABLESPACE_ENCRYPTION (see note above)	Oracle Database 19c (19.16) and later releases - indicates whether a new tablespace should be encrypted. Available options are AUTO_ENABLE, MANUAL_ENABLE, and DECRYPT_ONLY. Starting with Oracle Database 19c (19.16), Oracle Cloud forces encryption for all tablespaces in the cloud database. This cannot be overridden. To prevent encrypted tablespaces on an on-premises database (primary or standby) set the TABLESPACE_ENCRYPTION parameter to DECRYPT_ONLY. DECRYPT_ONLY is only valid in an on-premises database. See TABLESPACE_ENCRYPTION	N/A	N/A	RECOMMENDED

To configure TDE, follow the steps in Setting Up the Transparent Data Encryption Wallet in *Move to Oracle Cloud Using Zero Downtime Migration*.

Checking the TDE Master Key Before Instantiation

Even in cases where the primary database remains unencrypted, TDE must be configured on the primary database. This configuration includes creating the encryption wallet and setting the master key.

During the process the wallet is copied to the standby database. The master key stored in the wallet will be used by the standby database for encryption.

In the event of a switchover where the cloud standby database becomes the primary database, the key is used by the unencrypted on-premises database to decrypt the encrypted redo from the cloud database.

Failure to set the master key will result in failure of Data Guard managed recovery.

To confirm the master key is set properly:

- Verify that the `MASTERKEYID` column in `V$DATABASE_KEY_INFO` matches a key existing in `V$ENCRYPTION_KEYS` on the source database.

  In a multitenant container database (CDB) environment, check `CDB$ROOT` and all the PDBs except `PDB$SEED`.

Configuring Online Redo Logs

Redo log switches can have a significant impact on redo transport and apply performance. Follow these best practices for sizing the online redo logs on the primary database before instantiation.

- All online redo log groups should have identically sized logs (to the byte).
- Online redo logs should reside on high performing disks (DATA disk groups).
- Create a minimum of three online redo log groups per thread of redo on Oracle RAC instances.
- Create online redo log groups on shared disks in an Oracle RAC environment.
- Multiplex online redo logs (multiple members per log group) *unless* they are placed on high redundancy disk groups.
- Size online redo logs to switch no more than 12 times per hour (every ~5 minutes). In most cases a log switch every 15 to 20 minutes is optimal even during peak workloads.

Sizing Redo Logs
Size redo logs based on the peak redo generation rate of the primary database.

You can determine the peak rate by running the query below for a time period that includes the peak workload. The peak rate could be seen at month-end, quarter-end, or annually. Size the redo logs to handle the highest rate in order for redo apply to perform consistently during these workloads.

```
SQL> SELECT thread#,sequence#,blocks*block_size/1024/1024 MB,
(next_time-first_time)*86400 sec,
 blocks*block_size/1024/1024)/((next_time-first_time)*86400) "MB/s"
 FROM v$archived_log
 WHERE ((next_time-first_time)*86400<>0)
 and first_time between to_date('2015/01/15 08:00:00','YYYY/MM/DD HH24:MI:SS')
 and to_date('2015/01/15 11:00:00','YYYY/MM/DD HH24:MI:SS')
 and dest_id=1 order by first_time;

   THREAD#  SEQUENCE#         MB        SEC       MB/s
---------- ---------- ---------- ---------- ----------
```

```
2       2291 29366.1963        831  35.338383
1       2565 29365.6553        781 37.6000708
2       2292 29359.3403        537  54.672887
1       2566 29407.8296        813 36.1719921
2       2293 29389.7012        678 43.3476418
2       2294 29325.2217       1236 23.7259075
1       2567 11407.3379       2658 4.29169973
2       2295 29452.4648        477 61.7452093
2       2296 29359.4458        954 30.7751004
2       2297 29311.3638        586 50.0193921
1       2568 3867.44092       5510 .701894903
```

Choose the redo log size based on the peak generation rate with the following chart.

Peak Redo Rate	Recommended Redo Log Size
<= 1 MB/s	1 GB
<= 5 MB/s	4 GB
<= 25 MB/s	16 GB
<= 50 MB/s	32 GB
> 50 MB/s	64 GB

Creating the Target Database

The target database, which will become the standby database, is initially created by the Oracle Cloud automation. This approach ensures that the database is visible in the Oracle Cloud user interface and is available for a subset of cloud automation, such as patching.

> **Note:**
>
> Oracle Data Guard operations, such as switchover, failover, and reinstate, are manual operations performed with Data Guard Broker. Data Guard Life Cycle Management is not supported by the user interface in hybrid Data Guard configurations.

Once the database is created, the Zero Downtime Migration work flow removes the existing files and instantiates the standby database in its place.

The following are exceptions in a hybrid Data Guard configuration (as compared to Zero Downtime Migration) for the target database:

- The target database must use the **same** `db_name` as the source database.
- The target database must use a **different** `db_unique_name`.

Choosing an Instantiation Method

The two recommended options for a hybrid Data Guard standby instantiation with Zero Downtime Migration are direct data transfer and Object Storage Service.

- Direct data transfer - `DATA_TRANSFER_MEDIUM=DIRECT` - copies data files directly from the primary database using RMAN.
- Object Storage Service - `DATA_TRANSFER_MEDIUM=OSS` - performs a backup of the primary database to an OSS bucket and instantiates the standby database from the backup.

There are additional options for instantiating from an existing backup or an existing standby which are not covered by this procedure. See Using an Existing RMAN Backup as a Data Source and Using an Existing Standby to Instantiate the Target Database in *Move to Oracle Cloud Using Zero Downtime Migration* for details.

Setting Zero Downtime Migration Parameters

The Zero Downtime Migration physical migration response file parameters listed below are the key parameters to be set in most cases.

- `TGT_DB_UNIQUE_NAME` - The `db_unique_name` for the target cloud database as registered with clusterware (srvctl)
- `MIGRATION_METHOD=ONLINE_PHYSICAL` - Hybrid Data Guard setups all use `ONLINE_PHYSICAL` method
- `DATA_TRANSFER_MEDIUM=`**`DIRECT | OSS`** - `DIRECT` is not supported for source databases on versions earlier than Oracle 12.1
- `PLATFORM_TYPE=`**`EXACS | EXACC | VMDB`** - Choose the correct target Oracle Cloud platform to ensure proper configuration
- `HOST=`*`cloud-storage-REST-endpoint-URL`* - Required if using OSS data transfer medium
- `OPC_CONTAINER=`*`object-storage-bucket`* - Required if using OSS data transfer medium
- `ZDM_RMAN_COMPRESSION_ALGORITHM=BASIC`
- `ZDM_USE_DG_BROKER=TRUE` - Data Guard Broker is an MAA configuration best practice

If bastion hosts or other complexities are involved, see Setting Physical Migration Parameters in *Move to Oracle Cloud Using Zero Downtime Migration* for details.

Task 3: Instantiate the Standby Database

After the preparations are complete you can run a Zero Downtime Migration online physical migration job to instantiate the cloud standby database.

You will actually run two jobs using the Zero Downtime Migration commands in ZDMCLI: an evaluation job and the actual migration job.

1. Run the evaluation job.

   The evaluation job analyzes your topography configuration and migration job settings to ensure that the process will succeed when you run it against the production database.

   Use the `-eval` option in the `ZDMCLI migrate database` command to run an evaluation job, as shown here.

```
zdmuser> $ZDM_HOME/bin/zdmcli migrate database
 -sourcedb source_db_unique_name_value
 -sourcenode source_database_server_name
 -srcauth zdmauth
 -srcarg1 user:source_database_server_login_user_name
 -srcarg2 identity_file:ZDM_installed_user_private_key_file_location
 -srcarg3 sudo_location:/usr/bin/sudo
 -targetnode target_database_server_name
 -backupuser Object_store_login_user_name
 -rsp response_file_location
 -tgtauth zdmauth
 -tgtarg1 user:target_database_server_login_user_name
 -tgtarg2 identity_file:ZDM_installed_user_private_key_file_location
```

```
-tgtarg3 sudo_location:/usr/bin/sudo
-eval
```

There are more examples of the evaluation job options in Evaluate the Migration Job in *Move to Oracle Cloud Using Zero Downtime Migration*.

> **Note:**
>
> Because the hybrid Data Guard cloud standby instantiation process is a physical migration, the Cloud Premigration Advisor Tool (CPAT) is not supported.

2. Run the migration job.

   By default, Zero Downtime Migration performs a switchover operation immediately after the target database is instantiated, so the `-stopafter` option is used in the `ZDMCLI migrate database` command to stop the migration job after the standby database is created.

   Use the `-stopafter` option and set it to `ZDM_CONFIGURE_DG_SRC` as shown here.

```
zdmuser> $ZDM_HOME/bin/zdmcli migrate database
 -sourcedb source_db_unique_name_value
 -sourcenode source_database_server_name
 -srcauth zdmauth
 -srcarg1 user:source_database_server_login_user_name
 -srcarg2 identity_file:ZDM_installed_user_private_key_file_location
 -srcarg3 sudo_location:/usr/bin/sudo
 -targetnode target_database_server_name
 -backupuser Object_store_login_user_name
 -rsp response_file_location
 -tgtauth zdmauth
 -tgtarg1 user:target_database_server_login_user_name
 -tgtarg2 identity_file:ZDM_installed_user_private_key_file_location
 -tgtarg3 sudo_location:/usr/bin/sudo
 -stopafter ZDM_CONFIGURE_DG_SRC
```

   The job ID is shown in the command output when the database migration job is submitted. Save this information in case later diagnosis is required.

   There are more examples of the `ZDMCLI migrate database` command usage shown in Migrate the Database in *Move to Oracle Cloud Using Zero Downtime Migration*.

Task 4: Validate the Standby Database

When the Zero Downtime Migration job has stopped after the standby database is instantiated, validate the standby database.

Check the Oracle Data Guard Broker Configuration

Using the parameter `ZDM_USE_DG_BROKER=TRUE` in the Zero Downtime Migration response file creates a Data Guard Broker configuration. Data Guard Broker will be the primary utility to manage the life cycle operations for hybrid Data Guard configurations, because the Oracle Cloud user interface is not aware of the on-premises database.

Using DGMGRL, validate the Data Guard Broker configuration. Data Guard Broker commands listed can be run form the primary or standby database.

```
DGMGRL> show configuration

Configuration - ZDM_primary db_unique_name

  Protection Mode: MaxPerformance
  Members:
  primary db_unique_name - Primary database
    standby db_unique_name - Physical standby database

Fast-Start Failover:  Disabled

Configuration Status:
SUCCESS   (status updated 58 seconds ago)
```

Configuration Status should be `SUCCESS`. If any other status is shown, re-run the command after waiting 2 minutes to give the Broker time to update. If issues persist, see the Oracle Data Guard Broker documentation to diagnose and correct any issues.

Validate the Standby Database

Using DGMGRL, validate the standby database.

```
DGMGRL> validate database standby db_unique_name

  Database Role:     Physical standby database
  Primary Database:  primary db_unique_name

  Ready for Switchover:  Yes
  Ready for Failover:    Yes (Primary Running)

  Flashback Database Status:
    primary db_unique_name:  On
    standby db_unique_name:  Off  <- see note below

  Managed by Clusterware:
    primary db_unique_name:  YES
    standby db_unique_name:  YES
```

> **Note:**
>
> Steps to enable flashback database on the standby will be addressed in a future step.

Task 5: Implement Recommended MAA Best Practices

After standby instantiation, evaluate implementing the following Oracle MAA best practices to achieve better data protection and availability.

Key best practices are listed below. Also see Oracle Data Guard Configuration Best Practices for details about Oracle MAA recommended best practices for Oracle Data Guard.

Enable Flashback Database

Flashback Database allows reinstatement of the old primary database as a standby database after a failover. Without Flashback Database enabled, the old primary database would have to be recreated as a standby after a failover. If flashback database has not already been enabled, enable it now.

To enable flashback database, make sure you have sufficient space and I/O throughput in your Fast Recovery Area or RECO disk group, and evaluate any performance impact.

1. On the primary database, run the command below to enable flashback database on the primary if it is not already enabled.

```
SQL> alter database flashback on;
Database altered.
```

2. On the standby database, to enable flashback database, first disable redo apply, enable flashback database, then re-enable redo apply.

```
DGMGRL> edit database standby-database set state=apply-off;
Succeeded.

SQL> alter database flashback on;
Database altered.

DGMGRL> edit database standby-database set state=apply-on;
Succeeded.
```

Set CONTROL_FILES Parameter and Change Default Open Mode of Standby

An Oracle MAA best practice recommendation is to have only one control file when placed on a high redundancy disk group. All Oracle Cloud offerings use high redundancy, therefore only one control file is required.

1. On the standby database, edit the `CONTROL_FILES` parameter.

```
SQL> show parameter control_files

NAME                                 TYPE        VALUE
------------------------------------ -----------
------------------------------
control_files                        string      controlfile-1
                                                 , controlfile-2

SQL> ALTER SYSTEM SET control_files='controlfile-1' scope=spfile sid='*';
System altered.
```

2. Stop the database as the `oracle` user, and then, as the `grid` user, remove the extra control file (`su` to the `grid` user from the `opc` user).

```
$ srvctl stop database -db standby-unique-name
[grid@standby-host1 ~]$ asmcmd rm controlfile-2
```

3. While the database is down, modify the start option so the standby database default is open read only, and then start the database.

```
$ srvctl modify database -db standby-unique-name -startoption 'read only'
$ srvctl start database -db standby-unique-name
```

> **Note:**
>
> The Oracle MAA best practice is for the standby to be open read-only to enable Automatic Block Media Recovery; however, Oracle Cloud supports a mounted standby. If a mounted standby is your preferred configuration it can be configured.

Set Alternate Local Archive Log Location

In the event that space is exhausted in the recovery area, a primary database will stop archiving and all operations will halt until space is made available to archive the online redo logs.

To avoid this scenario, create an alternate local archive location on the DATA disk group.

1. Set `LOG_ARCHIVE_DEST_10` to use the DATA disk group and set the state to `ALTERNATE`.

```
SQL> ALTER SYSTEM SET log_archive_dest_10='LOCATION=+DATAC1
 VALID_FOR=(ALL_LOGFILES,ALL_ROLES) MAX_FAILURE=1
 REOPEN=5 DB_UNIQUE_NAME=standby-unique-name
 ALTERNATE=LOG_ARCHIVE_DEST_1' scope=both sid='*';

SQL> ALTER SYSTEM SET log_archive_dest_state_10=ALTERNATE scope=both
sid='*';
```

2. Set `LOG_ARCHIVE_DEST_1` to use `LOG_ARCHIVE_DEST_10` as an alternate.

```
SQL> ALTER SYSTEM SET
log_archive_dest_1='LOCATION=USE_DB_RECOVERY_FILE_DEST
 VALID_FOR=(ALL_LOGFILES,ALL_ROLES) MAX_FAILURE=1
 REOPEN=5 DB_UNIQUE_NAME=standby-unique-name
 ALTERNATE=LOG_ARCHIVE_DEST_10' scope=both sid='*';
```

> **Note:**
>
> When backups are not configured, by default archived logs older than 24 hours are swept every 30 minutes.

Set Data Protection Parameters

MAA best practice recommendations include the following settings on the primary and standby databases.

```
db_block_checksum=TYPICAL

db_lost_write_protect=TYPICAL
```

```
db_block_checking=MEDIUM

SQL> show parameter db_block_checksum

NAME                                 TYPE        VALUE
------------------------------------ -----------
------------------------------
db_block_checksum                    string      TYPICAL

SQL> alter system set db_block_checksum=TYPICAL scope=both sid='*';

SQL> show parameter db_lost_write_protect

NAME                                 TYPE        VALUE
------------------------------------ -----------
------------------------------
db_lost_write_protect                string      typical

SQL> alter system set db_lost_write_protect=TYPICAL scope=both sid='*';

SQL> show parameter db_block_checking

NAME                                 TYPE        VALUE
------------------------------------ -----------
------------------------------
db_block_checking                    string      OFF

SQL> alter system set db_block_checking=MEDIUM scope=both sid='*';
```

Note that the `db_block_checking` setting has an impact on primary database performance and should be thoroughly tested with a production workload in a lower, production-like environment.

If the performance impact is determined to be unacceptable on the primary database, the standby database should set `db_block_checking=MEDIUM` and set the `cloudautomation` Data Guard Broker property to '1' for both databases so that the value will be changed appropriately after a role transition.

```
DGMGRL> edit database primary-unique-name set property cloudautomation=1;
Property "cloudautomation" updated

DGMGRL> edit database standby-unique-name set property cloudautomation=1;
Property "cloudautomation" updated
```

Note that the cloudautomation property must be set on both databases to work properly.

Configure Redo Transport - Oracle Net Encryption

To protect against plain text or unencrypted tablespace redo from being visible on the WAN, place the following entries in the sqlnet.ora file on all on-premises and cloud databases.

Cloud deployments use the `TNS_ADMIN` variable to separate tnsnames.ora and sqlnet.ora in shared database homes. Therefore, the cloud sqlnet.ora, and by extension tnsnames.ora, for a given database are located in $ORACLE_HOME/network/admin/*db_name*.

These values should already be set by the deployment tool in cloud configurations.

```
SQLNET.ORA ON ON-PREMISES HOST(S)
SQLNET.ENCRYPTION_SERVER=REQUIRED
SQLNET.CRYPTO_CHECKSUM_SERVER=REQUIRED
SQLNET.ENCRYPTION_TYPES_SERVER=(AES256,AES192,AES128)
SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER=(SHA1)
SQLNET.ENCRYPTION_CLIENT=REQUIRED
SQLNET.CRYPTO_CHECKSUM_CLIENT=REQUIRED
SQLNET.ENCRYPTION_TYPES_CLIENT=(AES256,AES192,AES128)
SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT=(SHA1)
```

> **Note:**
>
> If **all** tablespaces and data files are encrypted with TDE, Oracle Net encryption is redundant and can be omitted.

Configure Redo Transport - Reconfigure Redo Transport Using Full Connect Descriptors

For simplicity, Zero Downtime Migration uses an EZconnect identifier to set up Oracle Data Guard redo transport.

For short lived configurations, like those with a full Zero Downtime Migration work flow, this solution is acceptable. However, for hybrid Data Guard configurations, the MAA best practice recommendation is to use a full connect descriptor configured in tnsnames.ora.

Use the following example, replacing attribute values with values relevant to your configuration.

The TNS descriptors for the databases will be different depending on whether the SCAN listeners are resolvable from the other system.

The description below assumes that the SCAN name is resolvable and can be used in the TNS descriptor. If a SCAN name cannot be resolved, an `ADDRESS_LIST` can be used. See Multiple Address Lists in tnsnames.ora for details.

Add the following descriptors to a shared tnsnames.ora file on the primary and standby database systems after making the appropriate replacements.

```
standby-db_unique_name =
    (DESCRIPTION=
      (ADDRESS=
        (PROTOCOL= TCP)
        (HOST= standby-cluster-scan-name )
        (PORT=standby-database-listener-port))
      (CONNECT_DATA=
        (SERVER= DEDICATED)
        (SERVICE_NAME= standby-database-service-name)))
primary-db_unique_name=
    (DESCRIPTION=
      (ADDRESS=
        (PROTOCOL=TCP)
        (HOST=primary-cluster-scan-name)
        (PORT=primary-database-listener-port))
      (CONNECT_DATA=
        (SERVER=DEDICATED)
```

```
    (SERVICE_NAME=primary-database-service-name)
  ))
```

> **Note:**
>
> A descriptor with the name of the primary `db_unique_name` may have been created by cloud automation or Zero Downtime Migration. Replace this entry, because it points to the wrong database.

Configure Redo Transport - Modify Data Guard Broker Settings for Redo Transport

Change the EZconnect identifier, which was set during the Zero Downtime Migration work flow, to use the connect descriptors added to the tnsnames.ora files for each database.

```
DGMGRL> show database primary-db_unique_name DGConnectIdentifier
DGConnectIdentifier = 'ZDM-created-EZconnect-string>'
DGMGRL> edit database primary-db_unique_name
         set property DGConnectIdentifier='primary-db_unique_name';
DGMGRL> show database standby-db_unique_name DGConnectIdentifier
DGConnectIdentifier = 'ZDM-created-EZconnect-string'
DGMGRL> edit database standby-db_unique_name
         set property DGConnectIdentifier='standby-db_unique_name';
```

Configure Standby Automatic Workload Repository

Standby Automatic Workload Repository (AWR) allows the AWR reports to be produced against the standby. These reports are very important when diagnosing redo apply and other performance issues on a standby database.

It is strongly recommended that you configure standby AWR for all Oracle Data Guard configurations.

See My Oracle Support note How to Generate AWRs in Active Data Guard Standby Databases (Doc ID 2409808.1) for information.

Configure Backups

Optionally, configure automatic backups for the Oracle cloud database, for primary or standby role, using Autonomous Recovery Service or object storage.

Autonomous Recovery Service allows you to offload backup processing and storage in addition to the following benefits:

- Significantly reduce dependencies on backup infrastructure
- Develop a centralized backup management strategy for all the supported OCI database services
- Retain backups with Recovery Service for a maximum period of 95 days
- Leverage real-time data protection capabilities to eliminate data loss
- Significantly reduce backup processing overhead for your production databases
- Implement a dedicated network for Recovery Service operations in each virtual cloud network (VCN)
- Automate backup validation to ensure recoverability

- Implement a policy-driven backup life-cycle management

See Manage Database Backup and Recovery on Oracle Exadata Database Service on Dedicated Infrastructure and Database Autonomous Recovery Service for more information.

Health Check and Monitoring

After instantiating the standby database, a health check should be performed to ensure that the Oracle Data Guard databases (primary and standby) are compliant with Oracle MAA best practices.

It is also recommended that you perform the health check monthly, and before and after database maintenance. Oracle Autonomous Health Framework and automated tools including an Oracle MAA Scorecard using OraChk or ExaChk are recommended for checking the health of a Data Guard configuration. See Oracle Autonomous Health Framework User's Guide and Oracle ORAchk and Oracle EXAchk documentation.

Regular monitoring of the Oracle Data Guard configuration is not provided in a hybrid Data Guard configuration and must be done manually. See Monitor an Oracle Data Guard Configuration for more information.

Part VII
Continuous Availability for Applications

33

Configuring Continuous Availability for Applications

Ensure that your applications are configured to quickly and automatically shift workload to available Oracle RAC instances or standby databases during planned maintenance and unplanned outages.

Application up time is maximized by following these recommendations when there are outages.

The primary audience for this document is application developers and application owners. Operational examples are included for database administrators and PDB administrators.

Topics:

About Application High Availability Levels

Depending on your application's high availability requirements, you can implement the level of high availability (HA) protection that you need.

HA protection levels are defined in the table below, and each increase in level builds upon the previous level.

HA Level	Configuration	Experience	Benefits
Level 1: Basic Application High Availability See Configuring Level 1: Basic Application High Availability	Database or Security Administrator: • Configure role-based database services • Leverage recommended database connection string, and optionally configure LDAP and wallets • Enable Fast Application Notification (FAN) for immediate interrupts during outages Application Developer: • Use MAA recommended connect string • Use Basic exception handling **Implementation effort:** Minimal - estimated 1 hour for administrator, and less than 1 hour for developers	Implementing Level 1 protection provides significant benefits compared to third party application failover solutions due to application + Oracle integration and intelligence to reduce application impact. • Reduced application downtime • Applications see errors during planned maintenance and unplanned outages, and automatically reconnect to another Oracle RAC instance or database with the target service • Applicable for unplanned outages and planned maintenance. In some cases, long running transactions should be deferred or suspended during planned maintenance.	High availability with application automatically failing over and reconnecting • Quick timeouts and automated connection retry with database connect string • Location transparency with services: role-based services for standby and read-only databases so that applications are automatically routed to the proper instance with the correct role • FAN auto-configured with database connect string • Immediate interrupt on outages when using FAN (no need to tune timeouts and wait for them) • Clusterware is aware of RAC and VIP health, so there is no waiting on downed end points thanks to FAN
Level 2: Prepare Applications for Planned Maintenance See Configuring Level 2: Prepare Applications for Planned Maintenance	Level 1 configuration + Application Developer: • Use Oracle connection pools or connection tests and return your connection to the pool between uses **Additional implementation effort for developers:** Minimal effort with Oracle Connection pools - up to hours of effort when using an application server (or days when not using an application server depending on application complexity) for developers to identify connection tests used in the application and possibly create new ones in the database	• Avoids errors during planned maintenance (errors still possible for unplanned outages) • Ability to drain and move workload gracefully without application interruption • Applicable for unplanned outages and planned maintenance events. In some cases, long running transactions should be deferred or suspended during planned maintenance.	Workload moves gracefully across instances with a slight delay and no errors during planned maintenance

HA Level	Configuration	Experience	Benefits
Level 3: Mask Unplanned and Planned Failovers from Applications See Configuring Level 3: Mask Unplanned and Planned Failovers from Applications	Level 1 and 2 configuration + "Application Continuity" Solution Database or Security Administrator: • Additional security and privileges required Application Developers: • External actions (for example, side effects) outside the database need to be considered **Additional implementation effort:** Days to weeks of collaboration between developers and database administrators to review protection coverage (depending on application complexity)	• In-flight transactions automatically acknowledge the commit or replay without application code changes • Database administrators and application developers coordinate to ensure readiness using AWR statistics to assess protection coverage, and use ACCHK (application continuity health check) to identify coverage or exceptions when transactions can or cannot be replayed	Masks unplanned and planned fail overs from applications • Applications avoid seeing errors during planned maintenance and outages • In-flight uncommitted transactions are replayed; committed transactions are acknowledged and not replayed

All of the HA Levels described in the table above are superior to connection management approaches using load balancers as single connection VIP endpoints for the following reasons:

- **Smart Service Health and Intelligent Reconnect:** Oracle Clusterware and Oracle Data Guard Broker closely monitor the health and state of the clusters and databases to ensure connections are routed to the database service that is opened on a primary.
- **Transparent and Automatic Failover:** There is no need to query the health of databases and decide which is the proper one to move a VIP; everything is transparent in the high availability approaches described in the table.
- **Fast Notification and Automatic Connection Retries:** The disconnection of already connected sessions is immediate, and happens intelligently when Oracle Clusterware and Data Guard Broker detect outages or role changes on the primary and standby databases.

Terms

The following terms are used throughout this document:

- **Draining:** Move a connection from one instance to another available instance.

  Draining to move sessions gracefully from one instance to another is used during planned maintenance and load rebalancing. The connection is moved when the application returns the connection to a pool and then obtains a new connection or another rule is satisfied.

- **Fail over:** Reestablish an equivalent session at a new instance that offers the service.

  Fail over occurs during unplanned outages and during planned maintenance when sessions do not drain within an allotted period of time. The application should not receive errors when Application Continuity is configured.

Software Recommendations

The following software is recommended for HA Level configurations:

- Oracle Real Application Clusters (Oracle RAC) and Oracle Clusterware (which provides services and infrastructure to efficiently manage outages), preferably with Oracle Grid Infrastructure (GI) release 19c or later
- Oracle Active Data Guard is recommended for protection from database, cluster, storage or site failures
- Oracle Database 19c client and database or a later long-term support version, with the most recent patch level

Configuring Level 1: Basic Application High Availability

Implement a level of high availability that allows applications to immediately react to instance, node, or database failures, and quickly establish new connections to surviving database instances.

With application HA Level 1, downtime is minimized for unplanned and planned outages. You get these benefits by ensuring that the application configuration implements these recommendations. No code changes are required.

At a high level, the steps to implement Level 1 are:

- Step 1: Configure High Availability Database Services
- Step 2: Configure the Connection String for High Availability
- Step 3: Ensure That FAN Is Used
- Step 4: Ensure Application Implements Reconnection Logic

Step 1: Configure High Availability Database Services

Create a non-default, role-based database service to use high-availability features.

A database service is a logical abstraction for managing workloads or a group of applications sharing similar SLAs or types of workloads (for example, OLTP vs. batch). Database services provide location transparency and hide complex aspects of the underlying system from the client.

Your application must connect to a non-default database service to use high-availability features. You must explicitly create a service (or several services as needed for different application workloads) instead of using the default database service or the default PDB service (that is, the service with the same name as the database or PDB).

On Oracle Autonomous Database, services are created for you using recommended attributes.

About Server-Side Configuration for Services

These services are configured by a database administrator to set up services through Oracle Clusterware.

When using Oracle Data Guard and standby databases, create services using the primary role to ensure that applications connect to the primary database for read/write operations, and standby role for services to optionally offload read-only and small infrequent writes to the standby database.

Services start and stop automatically after a Data Guard role transition (for example, switchover or failover) based on their roles.

Configure your services according to your architecture in one of the following sections:

- Configure High Availability Services
- Configure High Availability Services for Oracle Active Data Guard or Standby Roles

> **Note:**
>
> Services must be started so that they can be used after creating them. Use a command like this:
>
> ```
> $ srvctl start service -db mydb -service my_service
> ```

See also:

Using Oracle Services in *Oracle Real Application Clusters Administration and Deployment Guide*

Configure High Availability Services

Create a non-default, role-based database service to use high-availability features.

A service may be configured to direct connections to a single preferred instance, or alternatively, if the preferred instance is down, to an available instance. When a service is available only on one instance, it is called a *singleton service.* This allows you to isolate workloads among instances in a cluster.

You could also configure a service to put connections on multiple instances of a cluster, to spread work across all instances. Also, if one instance is down, connections can be made on the surviving instances.

There are other combinations where you can configure a subset of instances as "preferred" and another subset of instances as "available". These subsets provide for spreading load across some instances while isolating work from others (and still have instances available in case of a failure).

Example 1: Singleton Service

This example creates a singleton service called MyService for the primary role, where the connections are made on instance inst1, unless that instance is not available. If the instance is not available, connections are made on inst2. It also configures a default drain timeout of 300 seconds to wait for sessions to drain; at the end of that time any remaining sessions are terminated due to the `IMMEDIATE` option.

The settings for `commit_outcome` and `failovertype` enable Transparent Application Continuity (TAC) for the future, if you decide to implement it (this is an advanced feature; see Oracle Application Continuity on Oracle MAA for details). Enabling TAC has no detrimental impact and automatically provides benefits when prerequisites are met, if you should decide to move to HA Level 3.

```
$ srvctl add service -db mydb -service my_service -pdb mypdb
 -preferred inst1 -available inst2 -commit_outcome TRUE
 -failovertype AUTO -notification TRUE -drain_timeout 300
 -stopoption IMMEDIATE -role PRIMARY
```

If you want your application to gracefully switch to another Oracle RAC instance with no application blackout, set the `drain_timeout` interval to a sufficient timeout that allows your

applications to close their connections between transactions and gracefully stop or move to another instance. The `drain_timeout` interval is best leveraged for short OLTP applications. For large batch operations, it's best defer or suspend these operations before a planned maintenance window.

Example 2: Service with Multiple Instances

This example creates a service that is similar to the singleton above but spreads connections across multiple instances in this cluster:

```
$ srvctl add service -db mydb -service my_service -pdb mypdb
 -preferred inst1,inst2 -commit_outcome TRUE -failovertype AUTO
 -notification TRUE -drain_timeout 300 -stopoption IMMEDIATE
 -role PRIMARY
```

Configure High Availability Services for Oracle Active Data Guard or Standby Roles

Create a service used to connect to a standby database (read-only physical standby).

Create a service as shown in the following example:

```
$ srvctl add service -db mydb -service my_standby_service
 -pdb mypdb –preferred inst1 -available inst2 -notification TRUE
 -drain_timeout 300 -stopoption IMMEDIATE -role PHYSICAL_STANDBY
```

Step 2: Configure the Connection String for High Availability

Oracle recommends that your application use the connection string configuration shown here to connect successfully during various scenarios including database switchover and failover to other sites.

Example 1: Connect string with Oracle RAC primary database and no standby

```
Alias = (DESCRIPTION =
(CONNECT_TIMEOUT= 90)(RETRY_COUNT=20)(RETRY_DELAY=3)
(TRANSPORT_CONNECT_TIMEOUT=1000ms)
    (ADDRESS_LIST =
        (LOAD_BALANCE=on)
        (ADDRESS = (PROTOCOL = TCP)(HOST=clu_site1-scan)(PORT=1521)))
    (CONNECT_DATA=(SERVICE_NAME = my_service)))
```

Example 2: Connect string with Oracle RAC primary and standby databases

This example makes connections to an Oracle RAC primary database or a standby database, depending on which one is available.

```
Alias = (DESCRIPTION =
(CONNECT_TIMEOUT= 90)(RETRY_COUNT=100)(RETRY_DELAY=3)
(TRANSPORT_CONNECT_TIMEOUT=1000ms)
    (ADDRESS_LIST =
        (LOAD_BALANCE=on)
        (ADDRESS = (PROTOCOL = TCP)(HOST=clu_site1-scan)(PORT=1521)))
    (ADDRESS_LIST =
        (LOAD_BALANCE=on)
```

```
    (ADDRESS = (PROTOCOL = TCP)(HOST=clu_site2-scan)(PORT=1521)))
(CONNECT_DATA=(SERVICE_NAME = my_service)))
```

> **Note:**
>
> clu_site1-scan and clu_site2-scan refer to SCAN listeners in a cluster on site1 and site2, respectively.

It's recommended that you use the most recent drivers, but all Oracle drivers from release 12.2 and later should use the example connection strings above. Specific values can be tuned, but the values shown in this example are reasonable starting points, and so usable for almost all cases.

It is highly recommended that you maintain your connect string or URL in a central location, such as LDAP or `tnsnames.ora`. Do not scatter the connect string or URL in property files or private locations, as doing so makes it extremely difficult to maintain. Using a centralized location helps you preserve standard format, tuning, and service settings. Oracle's solution for this is to use LDAP with the Oracle Unified Directory product.

See also:

- Connection Time Estimates During Data Guard Switchover or Failover
- Oracle Net TNS String Parameters
- Oracle Unified Directory in *Administering Oracle Unified Directory*
- Overview of Local Naming Parameters in *Oracle Database Net Services Reference*

Step 3: Ensure That FAN Is Used

FAN provides an intelligent and immediate interrupt when outages occur allowing for a much smaller application impact or brownout.

When a service needs to drain for routine maintenance, or unplanned failures (such as node or network outages), the application needs to be informed in real time, so that it moves connections quickly to another instance or site. This is accomplished using Oracle's Fast Application Notification (FAN) feature. Enable FAN to prevent applications from hanging when physical failures, such as node, network, or site failures occur.

FAN uses Oracle Clusterware's Oracle Notification Service (ONS) to receive events from the cluster. ONS requires ports to be available between the client and the servers, and in some cases this requires a firewall port to be opened.

Registration to receive FAN events is enabled automatically when using the recommended service and connect string in steps 1 and 2 above.

The ONS port (by default, 6200) needs to be opened on all of your database servers, the firewall, and Oracle Active Data Guard nodes. For cloud environments, this step is very important, including for Oracle Autonomous Database on Dedicated Exadata Infrastructure (ADB-D), Exadata Database Service on Dedicated Infrastructure (ExaDB-D), and Oracle Exadata Database Service on Cloud@Customer (ExaDB-C@C), as shown in the example.

Enabling FAN for Clients

There are no application code changes to use FAN. FAN only requires an Oracle driver and the recommended database connect string.

FAN is auto-configured and is enabled out of the box. When connecting to the Oracle database, the database uses the URL or TNS connect string to auto-configure FAN at the client.

It is important to use the TNS formats shown in Step 2 for auto-configuration of FAN; using a different format syntax can prevent FAN from being auto-configured. To use FAN, you must connect to a database service (That you configured in Step 1) and you must be able to receive events from the Oracle Notification Service (ONS), which may require a port to be opened, as mentioned above.

FAN may also be configured manually using connection pool settings (see below), if needed.

See below for configuration requirements with different pool types.

JDBC FAN Requirements

For client drivers using UCP:

- Use the recommended connection URL/string (see above) for auto-configuration of ONS.
- Include JDBC JAR files `ojdbc8.jar` (or later), `ons.jar`, and `simplefan.jar` on the `CLASSPATH` (plus optional wallet jars, if needed: `osdt_cert.jar`, `osdt_core.jar`, and `oraclepki.jar`).
- Set the pool or driver property to enable Fast Connection Failover (for example, in UCP it is set for the `PoolDataSource` using `setFastConnectionFailoverEnabled(true)`).
- Disable auto-commit connection property (for example, in UCP it is disabled for the `PoolDataSource` using `setConnectionProperty(OracleConnection.CONNECTION_PROPERTY_AUTOCOMMIT, "false");`)
- For third-party JDBC pools, Oracle recommends using Universal Connection Pool (UCP) as a data source.
- Open port 6200 for ONS communication from your database server (6200 is the default port, a different port may have been chosen).

If you are not able to use the recommended connect URL/string, configure your clients manually by setting:

```
oracle.ons.nodes=Node01:6200, Node02:6200, Node03:6200
```

Additional settings might be needed when configuring manually. For example `walletfile` and `walletpassword`.

Connection pools other than UCP will have analogous requirements.

OCI FAN Requirements

- For Oracle Call Interface (OCI) clients:

  OCI clients embed FAN at the driver level so that all clients can use them regardless of the pooling solution.

  The database service must have the attribute "-notification TRUE" set

  If oraaccess.xml is in use, ensure that the *events* tag is TRUE:

```
<oraaccess> xmlns="http://xmlns.oracle.com/oci/oraaccess"
 xmlns:oci="http://xmlns.oracle.com/oci/oraaccess"
 schemaLocation="http://xmlns.oracle.com/oci/oraaccess
```

```
 http://xmlns.oracle.com/oci/oraaccess.xsd">
 <default_parameters>
 <events>true</events>
 </default_parameters>
</oraaccess>
```

- For ODP.Net clients

  Specify HA events in the connect string

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

See also:

Overview of Oracle Integrated Clients and FAN in *Oracle Real Application Clusters Administration and Deployment Guide*

Step 4: Ensure Application Implements Reconnection Logic

Applications should be written to catch connection failure exceptions and errors during database calls so they can obtain new connections and continue with new work.

For JDBC-based apps, the `SQLRecoverableException` can be caught to distinguish connection errors from typical application or SQL errors. If a connection error is caught, then a new connection should be obtained. This is simpler and more robust than checking for individual Oracle errors (which can adjust by Oracle Database release) in the `SQLException` class.

See also:

Connection Retry Logic Examples

Configuring Level 2: Prepare Applications for Planned Maintenance

Building on application HA Level 1: Basic Application High Availability, Level 2 adds session draining configuration for minimal application impact during planned maintenance.

After implementing Level 1, you are ready to implement a planned maintenance solution appropriate to your application from one of the choices below. You can use planned operations to relocate or stop services, or to switch over, allowing for graceful completion of the users' work.

The recommended approach, to avoid impacting applications, is to drain work in an Oracle RAC rolling fashion. Typically a period of time is allocated to perform the draining. Our recommended choice is to use Oracle connection pools that are integrated with FAN to initiate draining.

If you are unable to drain, an alternative approach is to drain work before maintenance starts.

Other choices can be used if you aren't able to use Oracle connection pools.

Employ the following practices to increase your application high availability to level 2:

- Recommended Option: Use an Oracle Connection Pool - Use an Oracle Connection Pool and return your connection to the pool between requests.

Alternatively, configure UCP with a Third-Party Connection Pool or a Pool with Request Boundaries.

- Alternate Option: Use Connection Tests - If you cannot use an Oracle connection pool, you can use connection tests.
- Leverage Server-Side Operations for Planned Maintenance
- Ensure that sufficient node capacity is available so that the load from one instance can be spread to other available instances without impacting the workload during a maintenance period.

Recommended Option: Use an Oracle Connection Pool

Using a FAN-aware Oracle connection pool is the recommended solution for managing planned maintenance.

Oracle pools provide full lifecycle management: draining, reconnecting, and rebalancing across nodes and sites. As the maintenance progresses and completes (for each instance or node), sessions are moved and rebalanced across instances. There is no impact to users when your application uses an Oracle Pool with FAN and returns connections to the pool between requests.

Supported Oracle Pools include:

- Universal Connection Pool (UCP)
- WebLogic Active GridLink
- Tuxedo
- OCI Session Pool
- ODP.NET Managed and Unmanaged providers
- Oracle Session Pool for Python

When using these pools, no application changes are needed other than ensuring that your connections are returned to the pool between requests.

It is a best practice that an application obtains a connection only for the time that it needs it, and then returns the connection to the pool as soon as it is finished making its database calls. Holding a connection instead of returning it to the pool prevents the pool from gracefully moving sessions to available instances, and it uses resources inefficiently, requiring many more connections than would otherwise be used. An application should, therefore, obtain a connection and then return that connection immediately after the work is complete. The connections are then available for later use by other threads, or your thread when needed again. Returning connections to a connection pool is a general recommendation regardless of how draining is implemented.

> **Note:**
>
> The syntax for obtaining and returning a connection varies by pool implementation. For example, in UCP you use the `getConnection()` method of the `PoolDataSource` object to obtain a connection, and the `close()` method to return it after you've done some work in the database.

Oracle Connection Pools validate a connection whenever a connection is borrowed to ensure that the connection can be used without any errors.

See Universal Connection Pool Developer's Guide

Alternate Option: Use Connection Tests

If you cannot use an Oracle Pool, then the Oracle client drivers 19c or Oracle Database 19c will drain the sessions for you.

When services are relocated or stopped, or there is a switchover to a standby site via Oracle Data Guard, the Oracle Database and Oracle client drivers are notified to look for safe places to release connections according to the following:

- Standard connection tests for connection validity (for example `isValid()` in JDBC)
- Custom SQL tests for connection validity

For custom batch applications, test the connection between batches. When the connection test fails, create or borrow another connection.

For third-party connection pools, enable connection tests offered by the vendor. When the connection test fails, the third-party pool will close the connection and allow you to borrow another one.

Note:

- When you use a connection test, the outcome of the connection test applies to that session only. Do not use connection tests to make general decisions about the instance and to make a determination to stop more than the session to which the test applies.
- Disable connection pool properties for flushing and destroying the pool on connection test failure when using Oracle WebLogic Server data sources.
- A monitor is functionality that makes a decision about the health of an instance. With FAN and Runtime Load Balancing such monitors are no longer needed and not susceptible to incorrect decisions. If you do want a monitor, SQL in that monitor must NOT be misinterpreted as a connection test for draining the application. There are a few ways to avoid this misinterpretation:
  - Disable a monitor's specific health query using the `dbms_app_cont_admin` package:

    ```
    dbms_app_cont_admin.disable_connection_test(dbms_app_cont_admin
    .sql_test,'SELECT COUNT(1) FROM DUAL');
    ```

    Here, the query used by the monitor, `'SELECT COUNT(1) FROM DUAL'`, is not considered a connection test. If there are any connection tests that also use this query, then they would be disabled and a different query would be needed.
  - Embed a comment into the monitor query to distinguish it from any of the registered connection tests:

    ```
    SELECT /* My Health monitor query */ COUNT(1) monitor FROM DUAL
    ```

Use Standard Connection Tests to Drain at the JDBC Thin Driver

For non-Oracle pools, to use connection tests with the JDBC thin driver, do the following steps.

1. Enable connection tests in your pool (implementation varies by third-party pool) and use the following test, `java.sql.Connection.isValid(int timeout)`
2. Set the Java system properties
   - `-Doracle.jdbc.fanEnabled=true`
   - `-Doracle.jdbc.defaultConnectionValidation=SOCKET` (in Oracle Database 19c the `isValid()` call is local to the client and will not require a trip to the database)

Use OCI Connection Tests to Drain at the OCI Driver

When using Oracle Call Interface (OCI) session pool, this connection check is done for you. When using the OCI driver directly, use `OCI_ATTR_SERVER_STATUS`. This is the only method that is a code change.

In your code, check the server handle when borrowing and returning connections to see if the session is disconnected. When the service is stopped or relocated, the value `OCI_ATTR_SERVER_STATUS` is set to `OCI_SERVER_NOT_CONNECTED`.

The following code sample shows you how to use `OCI_ATTR_SERVER_STATUS`.

```
ub4 serverStatus = 0
OCIAttrGet((dvoid *)srvhp, OCI_HTYPE_SERVER,
   (dvoid *)&serverStatus, (ub4 *)0, OCI_ATTR_SERVER_STATUS, errhp);
if (serverStatus == OCI_SERVER_NORMAL)
printf("Connection is up.\n");
else if (serverStatus == OCI_SERVER_NOT_CONNECTED)
 printf("Connection is down.\n");
/* Close connection and get a new one */
```

Use Connection Tests to Drain at the Oracle Database

The Oracle Database 19c can drain your sessions. When your connection test is executed during maintenance, the database closes your session and the application server closes the connection.

Use the view `DBA_CONNECTION_TESTS` to see the connection tests and rules that are enabled for you. If you are using a SQL-based connection test, use the same SQL that is enabled in the database (the same identical statement) at your connection pool or application server.

If you need additional connection tests, you can add, delete, enable, or disable connection tests for a service, a pluggable database, or non-container database.

For example:

```
SQL> EXECUTE
 dbms_app_cont_admin.add_sql_connection_test('SELECT COUNT(1) FROM DUAL');

SQL> EXECUTE
 dbms_app_cont_admin.enable_connection_test(dbms_app_cont_admin.sql_test,
   'SELECT COUNT(1) FROM DUAL');

SQL> SELECT * FROM DBA_CONNECTION_TESTS
```

Use the USERENV Function to Drain PL/SQL-Based Workloads

Use the function `USERENV` to know whether your session is in draining mode. For example, use this function to decide when to stop and acquire a new connection in the case of a long running PL/SQL loop that is processing records.

```
SQL> select SYS_CONTEXT('USERENV', 'DRAIN_STATUS') from dual ;

SYS_CONTEXT('USERENV','DRAIN_STATUS')

--------------------------------------------------------------------------------
-

DRAINING

SQL> select SYS_CONTEXT('USERENV', 'DRAIN_STATUS') from dual ;

SYS_CONTEXT('USERENV','DRAIN_STATUS')

--------------------------------------------------------------------------------
-

NONE
```

Leverage Server-Side Operations for Planned Maintenance

Server-side operations are required to manage connections for planned maintenance.

Note that services connected to the Oracle Database are configured with connection tests and a drain timeout specifying how long to allow for draining, and the `stopoption` (typically `IMMEDIATE`), that applies after the drain timeout expires. The `stop`, `relocate`, and `switchover` commands managed by SRVCTL include a `drain_timeout` and `stopoption` switch to override values set on the service if needed.

Oracle recommends configuring services with the required drain timeout applicable to that service, so they are used automatically during maintenance operations.

Maintenance commands are similar to the commands described in the examples in Server-Side Planned Maintenance Command Examples. You can use these commands to start draining. Include additional options, if needed, as described in My Oracle Support (MOS) Note: Doc ID 1593712.1. Oracle tools, such as Fleet Patching and Provisioning (FPP) use these commands as well.

Oracle Clusterware can start instances that are not currently running, but can run a service that requires that instance. Services that cannot be relocated or do not need relocation, are stopped. If a singleton service is defined with no other available instances, then it may incur complete downtime, which is expected behavior. It is better to have preferred instances and at least one available instance always defined.

After the maintenance is complete and the instance is restarted, no additional SRVCTL action is required because the Oracle Clusterware service attribute automatically determines where services will end up.

See also:

Server Draining Ahead of Planned Maintenance in *Oracle Real Application Clusters Administration and Deployment Guide*

Configuring Level 3: Mask Unplanned and Planned Failovers from Applications

Building on Level 1 and Level 2, Application Continuity is highly recommended to mask database interruptions from the applications and to handle timeouts and outages.

Database interruptions from the applications can include application workload that won't drain (planned failover). Application Continuity is enabled on the database service where it can be configured to operate in two modes, AC and TAC.

Application Continuity (AC)

Application Continuity hides outages, starting with Oracle Database 12.1 for JDBC thin applications, and Oracle Database 12.2.0.1 for OCI and ODP.NET applications with support for open-source drivers, such as Node.js, and Python, beginning with Oracle Database 19c.

Application Continuity rebuilds and recovers the session from a known point, which includes session states and transactional states, then it replays all interrupted in-flight work (if it is already committed, it is not replayed). When the replay is complete, the results are returned to the application as if no interruption occurred.

Application Continuity is recommended for OLTP applications using an Oracle connection pool. It is enabled on the database service through which the application is connecting to the database.

Transparent Application Continuity (TAC)

Starting with Oracle Database19c, Transparent Application Continuity (TAC) automatically tracks and records the session and transactional states, so the database session can be recovered and rebuilt following recoverable outages. This is done with no reliance on application knowledge or application code changes, allowing you to enable TAC for your applications.

Application transparency and failover are achieved by consuming the state-tracking information that captures and categorizes the session state usage as the application issues calls to the database. Set `FAILOVERTYPE` to `AUTO` on your service.

If you are not using an Oracle connection pool (as with SQL*PLUS), or you do not have knowledge about the application, then enable TAC on your database service.

Planned Failover with AC and TAC

Planned failover is failover that is invoked by the Oracle Database at points where the database decides that a session is replayable and is expected not to drain.

Planned failover is enabled by default when using AC or TAC. It improves situations where other draining methods are not active, for example, because FAN or connection tests are not configured.

Planned failover expedites maintenance by failing over early when replay is enabled.

For example, planned failover with TAC is the maintenance solution used with SQL*Plus.

See also:

- Ensuring Application Continuity and Restrictions and Other Considerations for Application Continuity in *Oracle Real Application Clusters Administration and Deployment Guide*
- https://database-heartbeat.com/category/application-continuity/ blog

Return Connections to the Connection Pool

Request boundaries are required for Application Continuity (AC) and are recommended for Transparent Application Continuity (TAC).

Using an Oracle connection pool, such as Universal Connection Pool (UCP) or OCI Session Pool, the request boundaries are automatically embedded in the session for you, without changing the application, at appropriate points. The application should return the connection to the Oracle connection pool when the unit of work, the database request, is completed in order to insert the end of request boundary. This also applies to using ODP.Net Unmanaged Provider, WebLogic Active GridLink, and RedHat.

Transparent Application Continuity (TAC), in addition, will discover request boundaries. The conditions for discovering such a boundary in Oracle Database 19c are:

- No transaction is in progress
- Cursors are returned to the statement cache or canceled (cursor does not remain open across transactions)
- No un-restorable session state exists (PL/SQL globals, OJVM, populated temporary tables)

Set FAILOVER_RESTORE on the Service

To restore your session state at failover, set the attribute `FAILOVER_RESTORE` on your database service.

An application can be written to change the database session state (using `ALTER SESSION` commands typically), and these states need to be in place if you want the work to be replayed after failover.

In the service configuration, use `FAILOVER_RESTORE LEVEL1` for Application Continuity, or `FAILOVER_RESTORE AUTO` for TAC. Following the application HA Level 1 steps, the service is created with `FAILVOERTYPE AUTO` which automatically sets `FAILOVER_RESTORE AUTO`.

The use of wallets is highly recommended; AC and TAC leverage wallets to ensure all modifiable database parameters are restored automatically with `FAILOVER_RESTORE`. Wallets are enabled for ADB-D and ADB-S and are the same as those used for database links.

See also:

Configuring a Keystore for FAILOVER_RESTORE in *Oracle Real Application Clusters Administration and Deployment Guide* to learn how to set up wallets for databases.

Restore Original Function Values During Replay

Oracle Database 19c keeps the values of `SYSDATE`, `SYSTIMESTAMP`, `SYS_GUID`, and sequence.`NEXTVAL`, `CURRENT_TIMESTAMP`, and `LOCALTIMESTAMP` for SQL during replay.

If you are using PL/SQL, then `GRANT KEEP` for application users, and use the `KEEP` clause for a sequence owner. When the `KEEP` privilege is granted, replay applies the original function result at replay.

```
SQL> GRANT KEEP DATE TIME to scott;
SQL> GRANT KEEP SYSGUID to scott;
SQL> GRANT KEEP SEQUENCE mySequence on mysequence.myobject to scott;
```

Side Effects

When a database request includes an external call from the database, such as sending MAIL or transferring a file, this is termed a side effect.

When replay occurs, there is a choice as to whether side effects should be replayed. Many applications want to repeat side effects such as journal entries, sending mail, and file writes. For Application Continuity, side effects are replayed, but can be programmatically avoided. Conversely, Transparent Application Continuity does not replay side effects.

JDBC Configuration

Use `oracle.jdbc.replay.OracleDataSourceImpl` in a standalone manner, or configure it as connection factory class for a Java connection pool (such as UCP) or a WebLogic AGL Server connection pool.

See Configuring the Data Source for Application Continuity in Oracle Universal Connection Pool Developer's Guide for information about enabling AC/TAC on UCP. You would configure the JDBC driver data source class `oracle.jdbc.replay.OracleDataSourceImpl` as the connection factory class on the UCP data source PoolDataSourceImpl.

Note that the exact data source and connection pool configuration is always specific to a particular vendor product, such as 3rd-party connection pool, framework, application server, container, for example.

Monitoring

Application Continuity collects statistics to monitor your protection levels.

These statistics are saved in the Automatic Workload Repository (AWR) and are available in Automatic Workload Repository reports. Review the statistics to determine the extent of protected calls or If the protected call count or protected time decreases. Use the ACCHK utility for details as to the cause.

See also:

Application Continuity Protection Check in *Oracle Real Application Clusters Administration and Deployment Guide*

Reference

Reference topics for Configuring Continuous Availability for Applications.

Topics:

- Oracle Net TNS String Parameters
- Connection Retry Logic Examples
- Server-Side Planned Maintenance Command Examples

Connection Time Estimates During Data Guard Switchover or Failover

The settings in the connect string allow for the following maximum times to connect during switchover or failover.

- Data Guard Switchover:

  RETRY_COUNT X RETRY_DELAY = 100 x 3 sec = 300 sec.

- Data Guard Failover:

  (3 SCANs x TRANSPORT_CONNECT_TIMEOUT) + (RETRY_COUNT x (RETRY_DELAY + (3 SCANS x TRANSPORT_CONNECT_TIMEOUT))) = (3 x 1) + (100 x (3 + (3 x 1)) = 3 + 600 = 603 sec

After Data Guard switchover / Data Guard failover to clu-site2, initial connections to clu-site2 take 3 seconds when clu-site1 is down (use of a connection pool helps mitigate this delay). When clu-site1 is reachable again (when it becomes a standby) connections are nearly instantaneous because the listener on the standby will answer immediately that the service is not there, prompting the client to connect to the other `ADDRESS_LIST`.

- If the switchover or failover completes much earlier than the maximum time, the application will experience less impact.
- Increase RETRY_COUNT if your system might take longer than 300 sec to complete a switchover or failover. If you need more time to complete a Data Guard switchover, then change RETRY_COUNT greater than 100.
- If you aren't using Oracle Clusterware, then your HOST address will not reference a SCAN VIP but a single VIP. This means that `TRANSPORT_CONNECT_TIMEOUT` must be set to higher or lower values to account for network latency.

Oracle Net TNS String Parameters

The parameters used in the connect string are explained here.

CONNECT_TIMEOUT

Applies when a connection to a listener address is attempted.

This setting represents the maximum time in which a connection using a specific `ADDRESS` endpoint has to complete. It includes the transport connection time and any other actions that must happen (redirection from SCAN VIP to listener VIP and finally to the foreground spawned process).

`CONNECT_TIMEMOUT` should be larger than `TRANSPORT_CONNECT_TIMEOUT`, otherwise `TRANSPORT_CONNECT_TIMEOUT` is effectively capped by `CONNECT_TIMEOUT`. When `TRANSPORT_CONNECT_TIMEOUT` is not specified, then `CONNECT_TIMEOUT` acts as the timeout for the entire connection attempt to an `ADDRESS` endpoint, both transport and final connection to the database foreground.

Oracle recommends the value for `CONNECT_TIMEOUT` be large enough to account for the value of `TRANSPORT_CONNECT_TIMEOUT`, in addition to potential delays that may occur when connecting to busy listeners and hosts. The value of 90 seconds in the example connect string is very generous and might need to be shortened in some cases. But, if it is too short, then the

setting could be counter-productive because it causes additional attempts that can also fail, and can introduce more unproductive workload on the servers to handle connection requests that might be prematurely abandoned.

RETRY_COUNT

If a connection attempt fails across all `ADDRESS_LISTS`, then additional attempts to connect beginning with the first `ADDRESS_LIST` will be made `RETRY_COUNT` times.

This is useful when a switchover or failover to a standby is in progress and the connection needs to keep trying until the operation is complete.

RETRY_DELAY

Seconds in between retry attempts.

A short amount of time is given to allow the new primary database time to open. This parameter is used with RETRY_COUNT to wait a reasonable amount of time to connect to a newly opened database.

It is better to have short retry delays with many retry counts so the connection can complete close to the time the primary database opened.

TRANSPORT_CONNECT_TIMEOUT=1000ms

Allow up to 1000 milliseconds to connect to the listener using TCP hosts in the ADDRESS. If no connection is made, then try the next ADDRESS.

When an Oracle RAC SCAN host name is used, each IP in the SCAN address is expanded internally into a separate ADDRESS string. Each ADDRESS is then attempted if a connection attempt fails.

Adjust this parameter for your environment to minimize the time spent waiting for the listener endpoint connection to succeed. If the time is too short, you might give up on a connection attempt too soon, causing more delays and potentially a failure to connect. If the time is too long, then if the listener endpoint is unreachable, you might spend too much time waiting before giving up and trying another endpoint.

The host names specify SCAN VIPs. They are always available when using a cluster. This means that if a node or network is down, the VIP sends an instant reply, so that when connecting, the next address is used immediately if the service is not offered at the VIP address.

LOAD_BALANCE=ON within ADDRESS_LIST

When a `HOST` inside an `ADDRESS` resolves to multiple addresses for Oracle RAC SCAN, then all of the addresses are tried in a random order.

If you set `LOAD_BALANCE=OFF`, then the order is the same every time, which might overburden one of the SCAN listeners, so its recommended to set it to `ON`.

Connection Retry Logic Examples

Reference code examples for reconnection logic.

See Step 4: Ensure Application Implements Reconnection Logic for more information.

Simple Retry (SANITY CHECK)

```
Connection jdbcConnection = getConnection();
int iterationCount = 0;
int maxIterations = 10;
for (int i = 0; i < maxIterations, i++)) {
    try {
        // apply the raise (DML + commit):
        giveRaiseToAllEmployees(jdbcConnection, i * 5);
        // no exception, the procedure completed:
        iterationCount++;
        Thread.sleep(1000);
    } catch (SQLRecoverableException recoverableException) {
        // Get a new connection only if the error was recoverable.
        System.out.println("SQLRecoverableException on iteration " +
iterationCount )
        System.out.println("DB Connection lost - will attempt to get a new
connection to continue with the other iterations")

        // IF its OK to lose this work and move onto the next
        // iteration you could now try to get a new connection
        // This depends on what the code is doing; in many use
        // cases you must stop working, in others you can proceed
        // after logging a message to a log file
        // In our example, we assume we can proceed with the rest
        // of the loop if possible.
        // Using Transaction Guard, we can know if the work
        // committed and move on safely (covered in another example).
        try {
            jdbcConnection.close(); // close old  connection:
            System.out.println("Connection closed - getting a new one")
            jdbcConnection = getConnection(); // reconnect to continue with
other iterations
        } catch (Exception ex) {
           System.out.println("Unable to close or get a new connection -
giving up")
           throw ex;
        }
    } catch (SQLException nonRecoverableException) {
       // This is not a recoverable exception, so give up
       System.out.println("SQL UN-recoverable exception...give up the rest of
the iterations")
       throw nonRecoverableException;
    }
}
```

Connection Retry Logic with Transaction Guard

```
Connection jdbcConnection = getConnection();
boolean isJobDone = false;
while (!isJobDone) {
    try {
        // apply the raise (DML + commit):
        giveRaiseToAllEmployees(jdbcConnection, 5);
        // no exception, the procedure completed:
```

```
            isJobDone = true;
        } catch (SQLRecoverableException recoverableException) {
            // Retry only if the error was recoverable.
            try {
                jdbcConnection.close(); // close old  connection:
            } catch (Exception ex) {} // pass through other exceptions
            Connection newJDBCConnection = getConnection(); // reconnect to allow
retry
            // Use Transacton Guard to force last request: committed or
uncommitted
            LogicalTransactionId ltxid
                = ((OracleConnection) jdbcConnection).getLogicalTransactionId();
            isJobDone = getTransactionOutcome(newJDBCConnection, ltxid);
            jdbcConnection = newJDBCConnection;
        }
    }

    void giveRaiseToAllEmployees(Connection conn, int percentage) throws
    SQLException {
        Statement stmt = null;
        try {
            stmt = conn.createStatement();
            stmt.executeUpdate("UPDATE emp SET sal=sal+(sal*" + percentage + "/
    100)");
        } catch (SQLException sqle) {
            throw sqle;
        } finally {
            if (stmt != null)
                stmt.close();
        }
        // At the end of the request we commit our changes:
        conn.commit(); // commit can succeed but the commit outcome is lost
    }

    /**
     * GET_LTXID_OUTCOME_WRAPPER wraps DBMS_APP_CONT.GET_LTXID_OUTCOME
     */
    private static final String GET_LTXID_OUTCOME_WRAPPER =
        "DECLARE PROCEDURE GET_LTXID_OUTCOME_WRAPPER(" +
        "  ltxid IN RAW," +
        "  is_committed OUT NUMBER ) " +
        "IS " +
        "  call_completed BOOLEAN; " +
        "  committed BOOLEAN; " +
        "BEGIN " +
        "  DBMS_APP_CONT.GET_LTXID_OUTCOME(ltxid, committed, call_completed); " +
        "  if committed then is_committed := 1;  else is_committed := 0; end if;
    " +
        "END; " +
        "BEGIN GET_LTXID_OUTCOME_WRAPPER(?,?); END;";

    /**
     * getTransactionOutcome returns true if the LTXID committed or false
    otherwise.
     * note that this particular version is not considering user call completion
     */
```

```
boolean getTransactionOutcome(Connection conn, LogicalTransactionId ltxid)
throws SQLException {
    boolean committed = false;
    CallableStatement cstmt = null;
    try {
        cstmt = conn.prepareCall(GET_LTXID_OUTCOME_WRAPPER);
        cstmt.setObject(1, ltxid); // use this starting in 12.1.0.2
        cstmt.registerOutParameter(2, OracleTypes.BIT);
        cstmt.execute();
        committed = cstmt.getBoolean(2);
    } catch (SQLException sqlexc) {
        throw sqlexc;
    } finally {
        if (cstmt != null)
            cstmt.close();
    }
    return committed;
}
```

Server-Side Planned Maintenance Command Examples

> **Note:**
>
> - If you are using these commands in scripts, you may find it helpful to include `wait = yes`.
> - The parameters, `-force -failover` cause the service to start on other available instances configured on each service.
> - For more details see Managing a Group of Services for Maintenance in *Oracle Real Application Clusters Administration and Deployment Guide*.

To stop all instances on a node (node1) with all associated services' configured `-drain_timeout` and `-stopoption` parameters.

```
srvctl stop instance -db myDB -node node1 -force -failover
 -role primary
```

To stop one instance (inst1) with all associated services' configured `-drain_timeout` and `-stopoption` parameters

```
srvctl stop instance -db myDB -instance inst1 -force -failover
 -role primary
```

Stop all instances with explicit draining parameters that override the parameters configured for associated services.

```
srvctl stop instance -db db_name -node node_name
 -stopoption IMMEDIATE -drain_timeout <#> -force -failover
```

Stop a service with explicit draining parameters.

```
srvctl stop service -db db_name -service service_name
 -instance instance_name -drain_timeout <#> -stopoption IMMEDIATE
 -force -failover
```

To stop a service named GOLD on an instance named inst1 (a given instance) with a 5 minute drain timeout and an IMMEDIATE stop option.

```
srvctl stop service -db myDB -service GOLD -instance inst1
 -drain_timeout 300 -stopoption IMMEDIATE -force -failover
```

Stop a Data Guard instance with explicit draining parameters.

```
srvctl stop instance -db db_name -node node_name
 -stopoption IMMEDIATE -drain_timeout <#> -force -failover
 -role primary
```

Relocate all services by database, node, or PDB.

```
srvctl relocate service -database db_unique_name
  -pdb pluggable_database
 {-oldinst old_inst_name [-newinst new_inst_name] |
                          -currentnode current_node
                         [-targetnode target_node]}
  -drain_timeout timeout -stopoption stop_option -force

srvctl relocate service -database db_unique_name
 -oldinst old_inst_name [-newinst new_inst_name]
 -drain_timeout timeout -stopoption stop_option
 -force

srvctl relocate service -database db_unique_name
 -currentnode current_node [-targetnode target_node]
 -drain_timeout timeout -stopoption stop_option
 -force
```

To switch over to a Data Guard secondary site with a wait timeout of 60 seconds, using Data Guard Broker.

```
SWITCHOVER TO dg_south WAIT 60
```

To switch over to Data Guard secondary site with a wait timeout from the services, using Data Guard Broker.

```
SWITCHOVER TO dg_south WAIT
```

Part VIII
Oracle Multitenant Best Practices

34

Overview of Oracle Multitenant Best Practices

Oracle Multitenant is Oracle's strategic product for database consolidation.

The benefits of the Oracle Multitenant architecture include:

- Access isolation between individual Pluggable Databases (PDBs) stored in the same Container Database (CDB)
- Ability to manage many databases with the simplicity of managing just one CDB that contains many PDBs. By backing up your CDB, updating the CDB software or setting up a standby CDB for disaster recovery, you are essentially reducing the complexity and steps by administrating one CDB instead of applying the same administrative steps on many independent databases. You reduce administrative tasks, steps and errors.
- Sharing of system resources to reduce CAPEX with the flexibility to set resource limits for things like memory, I/O and on a per PDB level
- Flexibility to operate on an individual PDB, for example relocating a single PDB into another container and upgrading just that PDB
- Rapid cloning and provisioning
- Tight integration with Oracle RAC

The following table highlights various Oracle Multitenant configuration and operational best practices.

Table 34-1 Oracle Multitenant configuration and operational best practices

Use Case	Best Practices
Pluggable Database (PDB) configuration	For all Oracle RDBMS releases 12c Release 2 (12.2) to 21c, configure the CDB with local undo mode See Undo Modes in 12.2 Multitenant Databases - Local and Shared Modes (Doc ID 2169828.1)

Table 34-1 (Cont.) Oracle Multitenant configuration and operational best practices

Use Case	Best Practices
PDB service management	Mandatory MAA best practices for any Oracle databases with Oracle Clusterware (for example, Oracle RAC and single instance databases with Oracle Clusterware installed) 1. Never use PDB default services, nor SAVED STATE (except during relocate operations), nor database triggers to manage role-based services. 2. Use clusterware-managed distinct services per PDB for your application service, and leverage that application service to connect to the database. 3. When defining a clusterware-managed application service, define which PDB and services will be started and in which RAC instance and database role. 4. For Data Guard, always use role-based services by assigning a role to each clusterware-managed service. If the above practices are applied, you will have predictable service management during PDB open and Data Guard role transitions. This will lead to higher application service availability and avoid application errors. For single instance databases without MAA-recommended Oracle clusterware setup, follow these practices: 1. Never use PDB default services. 2. Use distinct services per PDB for your application service, and leverage that application service to connect to the database. 3. For non-Data Guard, use only SAVED state to open PDBs and start up explicit application services **OR** for Data Guard, use only AFTER STARTUP database triggers to programmatically manage which application services should be started depending on primary, READ ONLY, or snapshot standby database role. See Best Practices for Pluggable Database End User and Application Connection and Open on Database Startup (Doc ID 2833029.1)

Table 34-1 (Cont.) Oracle Multitenant configuration and operational best practices

Use Case	Best Practices
Using Data Guard with Oracle Multitenant	The following My Oracle Support notes provide operational best practice recommendations when using Oracle Multitenant in an Oracle Data Guard configuration. • Data Guard Impact on Oracle Multitenant Environments (Doc ID 2049127.1) • Making Use Deferred PDB Recovery and the STANDBYS=NONE Feature with Oracle Multitenant (Doc ID 1916648.1) during use cases: PDB creation, PDB migration, and PDB cloning • Reusing the Source Standby Database Files When Plugging a PDB into the Primary Database of a Data Guard Configuration (Doc ID 2273829.1) for PDB migration • Reusing the Source Standby Database Files When Plugging a non-CDB as a PDB into the Primary Database of a Data Guard Configuration (Doc ID 2273304.1) for PDB migration • Using standby_pdb_source_file_dblink and standby_pdb_source_file_directory to Maintain Standby Databases when Performing PDB Remote Clones or Plugins (Doc ID 2274735.1) for PDB remote clone or PDB plug-in • Parameter enabled_pdbs_on_standby and STANDBYS Option With Data Guard Subset Standby (Doc ID 2417018.1) to support subset standby
Data Guard: PDB switchover and failover use cases	Using Data Guard Broker to Migrate a Pluggable Database to a New Data Guard Configuration Document 2887844.1 PDB Failover in a Data Guard environment: Using Data Guard Broker to Unplug a Single Failed PDB from a Standby Database and Plugging into a New Container or Migrate a Single PDB into a New Container Document 2088201.1
PDB migration	The following My Oracle Support notes provide operational best practices for different types of PDB migration with minimal downtime: • Use Zero Downtime Migration if your target is Exadata platform or cloud. See https://www.oracle.com/database/technologies/rac/zdm.html • Step by Step Process of Migrating non-CDBs and PDBs Using ASM for File Storage (Doc ID 1576755.1) • Cloning a Pluggable Database from an RMAN Container Database Backup (Doc ID 2042607.1) • Using Data Guard Broker to Migrate a Pluggable Database to a New Data Guard Configuration Document 2887844.1
PDB relocation	• Using PDB Relocation to Upgrade an Individual PDB (Doc ID 2771716.1) • Using PDB Relocation to Move a Single PDB to Another CDB Without Upgrade (Doc ID 2771737.1)
PDB resource management	The following My Oracle Support note provides operational use cases for Oracle Multitenant resource management: How to Control and Monitor the Memory Usage (Both SGA and PGA) Among the PDBs in Mutitenant Database- 12.2 New Feature (Doc ID 2170772.1)

With Oracle Multitenant MAA solutions, you can achieve administration and system resource savings while still benefiting from various MAA solutions. The following tables highlight zero and near-zero downtime and data loss for various unplanned outages and planned maintenance activities.

Table 34-2 Unplanned Outages

Unplanned Outages	Key Features for Solution	RTO	RPO
Recoverable node or instance failures	Real Application Cluster (RAC) Application Continuity (AC)	Seconds	Zero
Database, cluster, and site failures	Active Data Guard fast-start failover	<2 Minutes	Zero or Seconds
Data corruptions	Active Data Guard that includes auto block repair of physical corruptions	Zero	Zero
PDB unrecoverable failure or "sick" PDB	PDB failover using Data Guard migrate command Another target CDB on the same cluster is required See PDB Failover in a Data Guard environment: Using Data Guard Broker to Unplug a Single Failed PDB from a Standby Database and Plugging into a New Container or Migrate a Single PDB into a New Container (Doc ID 2088201.1)	<2 Minutes	Zero or Seconds
PDB failover to active replica	Option 1: Failover entire CDB with Primary and Standby CDB Data Guard architecture Option 2: Create PDB replica with Oracle GoldenGate. Perform PDB active failover using PDB replica in a different CDB. Use Global Data Services and practices from Application Checklist for Continuous Service for MAA Solutions help with the application failover	Potentially Zero	Zero or Seconds

Table 34-3 Planned Maintenance

Planned Downtime	Solution	RTO
Software and hardware updates	Real Application Cluster (RAC) Application Checklist for Continuous Service for MAA Solutions	Zero
Major database upgrade for entire CDB	Active Data Guard DBMS_ROLLING	Seconds
Major database upgrade for single PDB within CDB	PDB Relocate + Upgrade See Using PDB Relocation to Upgrade an Individual PDB (Doc ID 2771716.1)	Minutes

Table 34-3 (Cont.) Planned Maintenance

Planned Downtime	Solution	RTO
Migration to remote CDB	PDB Relocate See Using PDB Relocation to Move a Single PDB to Another CDB Without Upgrade (Doc ID 2771737.1)	Minutes
Migration to remote CDB (logical migration)	Data Pump and Oracle GoldenGate or Zero Downtime Migration	Potentially Zero

35

PDB Switchover and Failover in a Multitenant Configuration

The use cases documented here demonstrate how to set up single pluggable database (PDB) failover and switchover for an Oracle Data Guard configuration with a container database (CDB) with many PDBs.

With Oracle Multitenant and the ability to consolidate multiple pluggable databases (PDBs) in a container database (CDB), you can manage many databases that have similar SLAs and planned maintenance requirements with fewer system resources, and more importantly with less operational investment. Leveraging Oracle Multitenant and its CDBs/PDBs technologies with Oracle's resource management, it is an effective means to reduce overall hardware and operational costs.

Planning and sizing are key prerequisites in determining which databases to consolidate in the same CDB. For mission critical databases that require HA and DR protection and minimal downtime for planned maintenance, it's important that you

- Size and leverage resource management to ensure sufficient resources for each PDB to perform within response and throughput expectations
- Target PDB databases that have the same planned maintenance requirements and schedule
- Target PDB databases that can all fail over to same CDB standby in case of unplanned outages such as CDB, cluster, or site failures

Note that Data Guard failover and switchover times can increase as you add more PDBs and their associated application services. A good rule is to have fewer than 25 PDBs per CDB for mission critical "Gold" CDBs with Data Guard if you want to reduce Data Guard switchover and failover timings.

Separating mission critical databases and dev/test databases into different CDBs is important. For example a mission critical "Gold" CDB with a standby may have only 5 PDBs with identical HA/DR requirements and may be sized to have ample system resource headroom while an important CDB with standby can contain 100 PDBs for dev, UAT and application testing purposes and may set up some level of over subscription to reduce costs. Refer to Overview of Oracle Multitenant Best Practices for more information on Multitenant MAA and Multitenant best practices.

This use case provides an overview and step by step instructions for the exception cases where a complete CDB Data Guard switchover and failover operation is not possible. With PDB failover and switchover steps, you can isolate the Data Guard role transition to one PDB to achieve Recovery Time Objective (RTO) of less 5 minutes and zero or near zero Recovery Point Objective (RPO or data loss).

Starting with Oracle RDBMS 19c (19.15) you can use Data Guard broker command line interface (DGMGRL) to migrate PDBs from one Data Guard configuration to another. Using broker, you can initiate PDB disaster recovery (DR) and switchover operations in isolation without impacting other PDBs in the same CDB.

The following primary use cases are described below for Data Guard broker migration:

- PDB switchover use case - Planned maintenance DR validation which invokes a PDB switchover operation without impacting existing PDBs in a Data Guard CDB
- PDB Failover use case - Unplanned outage DR which invokes a PDB failover without impacting existing PDBs in a Data Guard CDB

> **Note:**
>
> To relocate a single PDB when upgrade is not required without impacting other PDBs in a CDB see Using PDB Relocation to Move a Single PDB to Another CDB Without Upgrade (Doc ID 2771737.1). To relocate a single PDB requiring upgrade without impacting other PDBs in a CDB see .

PDB Switchover Use Case

In this PDB switchover or "DR Test" use case, a PDB is migrated from one Oracle Data Guard protected CDB to another Data Guard protected CDB.

As part of this use case, the files for the PDB on both the primary and standby databases of the source CDB are used directly in the respective primary and standby databases of the destination CDB.

The source CDB contains multiple PDBs, but we perform role transition testing on only one PDB because the others are not able to accept the impact. Before starting the migration, a second CDB must be created and it must have the same database options as the source CDB. The destination CDB is also in a Data Guard configuration, but it contains no PDBs at the start. The two corresponding primary and standby databases share the same storage and no data file movement is performed.

Prerequisites

Make sure your environment meets these prerequisites for the use case.

The Oracle Data Guard broker CLI (DGMGRL) supports maintaining configurations with a single physical standby database.

Using the method described here, for the PDB being migrated (the source), the data files of both the primary and the standby databases physically remain in their existing directory structure at the source and are consumed by the destination CDB and its standby database.

- Oracle patches/versions required
  - Oracle RDBMS 19c (19.15) or later
  - Patch 33358233 installed on the source and destination CDB RDBMS Oracle Homes to provide the broker functionality to manage the switchover process. You don't need to apply the patch on Oracle RDBMS 19c (19.18) and later; it is included.
  - Patch 34904997 installed on the source and destination CDB RDBMS Oracle Homes to provide the functionality to migrate the PDB back to the original configuration after performing the PDB Failover Use Case.
- Configuration
  - `DB_CREATE_FILE_DEST = ASM_Disk_Group`
  - `DB_FILE_NAME_CONVERT=""`

  - STANDBY_FILE_MANAGEMENT=AUTO
  - The source and destination standby CDBs must run on the same cluster
  - The source and destination primary CDBs should run from the same Oracle Home, and the source and destination standby CDBs should run from the same Oracle Home
  - The source and destination primary CDBs must run on the same host
  - The source and destination primary databases must use the same ASM disk group, and the source and destination standby databases must use the same ASM disk group
- You must have access to the following
  - Password for the destination CDB sysdba user
  - Password for the standby site ASM sysasm user (to manage aliases)
  - Password for the destination CDB Transparent Data Encryption (TDE) keystore if TDE is enabled

> **Note:**
>
> PDB snapshot clones and PDB snapshot clone parents are not supported for migration or failover.
> For destination primary databases with multiple physical standby databases you must either use the manual steps in Reusing the Source Standby Database Files When Plugging a PDB into the Primary Database of a Data Guard Configuration (Doc ID 2273829.1), or use the ENABLED_PDBS_ON_STANDBY initialization parameter in the standby databases, to limit which standby will be managed by this process. See Creating a Physical Standby of a CDB in *Oracle Data Guard Concepts and Administration* for information about using ENABLED_PDBS_ON_STANDBY.
>
> Existing ASM aliases for the source PDB migrated are managed by the broker during the migrate process. ASM only allows one alias per file, so existing aliases pointing to a different location must be removed and new ones in the correct location created.

Configuring PDB Switchover

You configure the "DR test" PDB switchover use case in the following steps.

The sample commands included in the steps below use the following CDB and PDB names.

- CDB100 (source CDB)
  - Contains PDB001, PDB002, PDB003; PDB001 will be configured for switchover
- CDB100_STBY (source standby CDB)
- CDB200 (destination CDB)
- CDB200_STBY (destination standby CDB)

Step 1: Extract PDB Clusterware managed services on the source database

Determine any application and end user services created for the source PDB that have been added to CRS.

Because there are certain service attributes such as database role not stored in the database, the detail attributes should be retrieved from CRS using SRVCTL CONFIG SERVICE.

1. Retrieve the service names from the primary PDB (PDB001 in our example).

```
PRIMARY_HOST $ sqlplus sys@cdb100 as sysdba
SQL> alter session set container=pdb001;
SQL> select name from dba_services;
```

2. For each service name returned, retrieve the configuration including `DATABASE_ROLE`.

```
PRIMARY_HOST $ srvctl config service -db cdb100 -s SERVICE_NAME
```

Step 2: Create an empty target database

Create an empty CDB (CDB200 in our example) on the same cluster as the source CDB (CDB100) which will be the destination for the PDB.

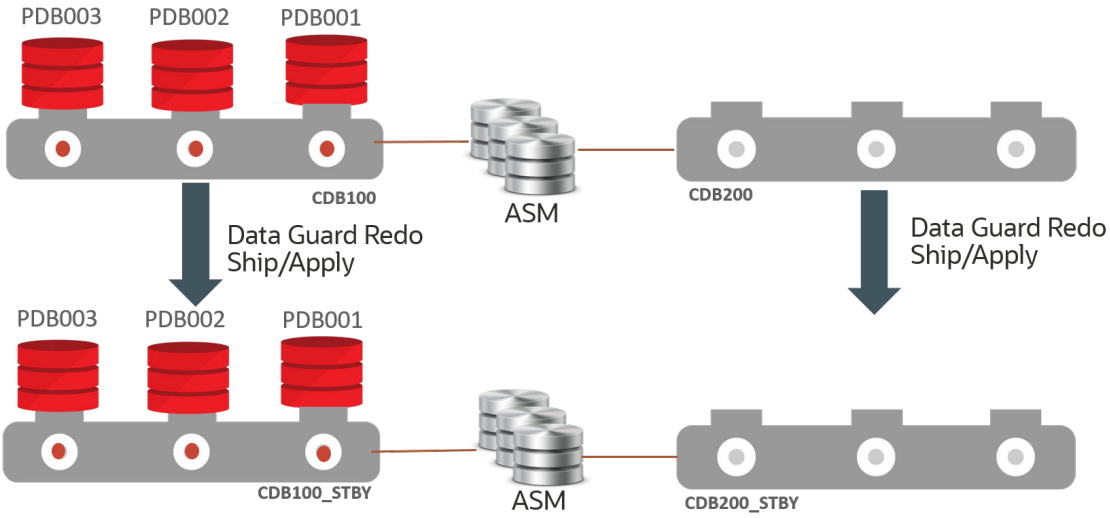
Allocate resources for this CDB to support the use of the PDB for the duration of the testing.

Step 3: Create a target standby database

Enable Oracle Data Guard on the target CDB to create a standby database (CDB200_STBY).

The standby database must reside on the same cluster as the source standby database.

The configuration should resemble the image below.



Step 4: Migrate the PDB

Migrate the PDB (PDB001) from the source CDB (CDB100) to the destination CDB (CDB200).

1. Start a connection to the source primary database using Oracle Data Guard broker command line (DGMGRL).

   This session should run on a host that contains instances of both the source primary CDB and the destination primary CDB. The session should be started with a sysdba user.

The broker CLI should be run from the command line on the primary CDB environment and run while connected to the source primary CDB. If you are using a TNS alias to connect to the source primary, it should connect to the source primary instance running on the same host as the broker CLI session.

The host and environment settings when running the broker CLI must have access to SQL*Net aliases for:

- Destination primary CDB – This alias should connect to the destination primary instance that is on the same host as the broker CLI session/source primary database instance to ensure the plug-in operation can access the PDB unplug manifest file that will be created.
- Destination standby CDB, this can connect to any instance in the standby environment.
- Standby site ASM instance, this can connect to any instance in the standby environment.

```
PRIMARY_HOST1 $ dgmgrl sys@cdb100_prim_inst1 as sysdba
```

This session should run on a host that contains instances of both the source primary CDB and the destination primary CDB and connected to a sysdba user. Use specific host/instance combinations instead of SCAN to ensure connections are made to the desired instances.

2. Run the `DGMGRL MIGRATE PLUGGABLE DATABASE` command.

   The `STANDBY FILES` keyword is required.

   See Full Example Commands with Output for examples with complete output and MIGRATE PLUGGABLE DATABASE for additional information about the command line arguments.

   - Sample command example without TDE:

```
DGMGRL> MIGRATE PLUGGABLE DATABASE PDB001 TO CONTAINER CDB200
 USING '/tmp/PDB001.xml' CONNECT AS sys/password@cdb200_inst1
 STANDBY FILES sys/standby_asm_sys_password@standby_asm_inst1
 SOURCE STANDBY CDB100_STBY DESTINATION STANDBY CDB200_STBY ;
```

   - Sample command example with TDE

```
DGMGRL> MIGRATE PLUGGABLE DATABASE PDB001 TO CONTAINER CDB200
 USING '/tmp/pdb001.xml' CONNECT AS sys/password@cdb200_inst1
 SECRET "some_value" KEYSTORE IDENTIFIED BY
"destination_TDE_keystore_passwd"
 STANDBY FILES sys/standby_asm_sys_password@standby_asm_inst1
 SOURCE STANDBY cdb100_stby DESTINATION STANDBY cdb200_stby;
```

When the command is executed, it:

1. Connects to the destination database and ASM instances to ensure credentials and connect strings are correct
2. Performs a variety of prechecks - If any precheck fails, the command stops processing and returns control to the user, an error is returned, and no changes are made on the target CDB

3. Creates a flashback guaranteed restore point in the destination standby CDB - This requires a short stop and start of redo apply
4. Closes the PDB on the source primary
5. Unplugs the PDB on the source primary - If TDE is in use, the keys are included in the manifest file generated as part of the unplug operation
6. Drops the PDB on the source primary database with the `KEEP DATAFILES` clause, ensuring that the source files are not dropped
7. Waits for the drop PDB redo to be applied to the source standby database - It must wait because the files are still owned by the source standby database until the drop redo is applied

   The command waits a maximum of `TIMEOUT` minutes (default 10). If the redo hasn't been applied by then the command fails and you must manually complete the process.
8. Manages the ASM aliases for the PDB files at the standby, removing any existing aliases and creating new aliases as needed - If the standby files already exist in the correct location, all aliases for the standby copy of the PDB are removed
9. Plugs in the PDB into the destination primary CDB - If TDE is in use, the keys are imported into the destination primary keystore as part of the plug-in
10. Ships and applies redo for the plug-in operation to the destination CDB, which uses any created aliases (if necessary) to access the files and incorporate them into the standby database
11. Validates that the standby files are added to the destination standby using redo apply
12. Opens the PDB in the destination primary database
13. Stops redo apply
14. Drops the flashback guaranteed restore point from the destination standby database
15. If TDE is enabled, redo apply remains stopped, if TDE is not enabled, redo apply is restarted

Step 5: Post Migration - Optional TDE Configuration Step and Restart Apply

If TDE is in use, redo apply will have been stopped by the broker `MIGRATE PLUGGABLE DATABASE` operation on the destination standby (CDB200_STBY) to allow the new TDE keys to be managed. Copy the keystore for the destination primary (CDB200) to the destination standby keystore and start redo apply.

```
SOURCE_HOST $ scp DESTINATION_PRIMARY_WALLET_LOCATION/*>
 DESTINATION_HOST:DESTINATION_STANDBY_WALLET_LOCATION/

$ dgmgrl sys/password@CDB200 as sysdba
DGMGRL> edit database cdb200_stby set state='APPLY-ON';
```

Step 6: Post Migration - Enable Services

Add any application services for the PDB to Cluster Ready Services (CRS), associating them with the PDB and correct database role in the destination CDB, and remove the corresponding service from the source CDB.

1. For each service on both the primary and standby environments, run the following:

   ```
   PRIMARY_HOST $ srvctl add service -db cdb200 -s SERVICE_NAME
    -pdb pdb001 -role [PRIMARY|PHYSICAL_STANDBY]….
   ```

```
STANDBY_HOST $ srvctl add service -db cdb200_stby -s SERVICE_NAME
 -pdb pdb001 -role [PRIMARY|PHYSICAL_STANDBY]….

PRIMARY_HOST $ srvctl remove service -db cdb100 -s SERVICE_NAME
STANDBY_HOST $ srvctl remove service -db cdb100_stby -s SERVICE_NAME
```

2. Start the required services for the appropriate database role.
   a. Start each PRIMARY role database service

      ```
      PRIMARY_HOST $ srvctl start service -db cdb200 -s SERVICE_NAME
      ```

   b. Start each `PHYSICAL_STANDBY` role database service:

      ```
      STANDBY_HOST $ srvctl start service -db cdb200_stby -s SERVICE_NAME
      ```

Step 7: Perform PDB role transition testing

After completing the migration, you can now perform required Oracle Data Guard role transitions or DR test with the PDB in destination CDB (CDB200). No other PDBs in the source CDB (CDB100) are impacted.

In addition, you continue to maintain the Data Guard benefits for both the source and destination CDB, such as DR readiness, Automatic Block Media Recovery for data corruptions, Fast-Start Failover to bound recovery time, Lost Write detection for logical corruptions, offloading reads to the standby to reduce scale and reduce impact of the primary, and so on. etc.

- Connect to the destination CDB using `DGMGRL` and perform the switchover.

```
$ dgmgrl sys@cdb200 as sysdba
DGMGRL> switchover to CDB200_STBY;
```

You can continue performing your DR testing for the PDB.

Once the DR testing for the PDB is complete, you can switch back and subsequently migrate the PDB back to the original source CDB.

- Connect to the destination CDB (CDB200) using `DGMGRL` and perform the switchback operation.

```
$ dgmgrl sys@cdb200 as sysdba
DGMGRL> switchover to CDB200;
```

Step 8: Return the PDB to the original CDB

After migration and role transition testing, switch back to the original configuration for this CDB and migrate the PDB back to the original Data Guard configuration, again automatically maintaining the standby database files. Data Guard broker migration handles any aliases that need to be dropped or created as part of the migration process.

See Full Example Commands with Output for examples with complete output.

- Start a connection to the source primary using Data Guard Broker command line (`DGMGRL`)

```
$ dgmgrl
DGMGRL> connect sys/@cdb200_inst1 as sysdba
```

– Command without TDE

```
DGMGRL> MIGRATE PLUGGABLE DATABASE PDB001 TO CONTAINER CDB100
 USING '/tmp/PDB001_back.xml' CONNECT AS sys/password@cdb100_inst1
 STANDBY FILES sys/standby_asm_sys_password@standby_asm_inst
 SOURCE STANDBY CDB200_STBY DESTINATION STANDBY CDB100_STBY ;
```

– Command with TDE

```
DGMGRL> MIGRATE PLUGGABLE DATABASE PDB001 TO CONTAINER CDB100
 USING '/tmp/PDB001_back.xml' CONNECT AS sys/password@cdb100_inst1
  SECRET "some_value" KEYSTORE
 IDENTIFIED BY "destination_TDE_keystore_passwd"
 STANDBY FILES sys/standby_asm_sys_password@standby_asm_inst
 SOURCE STANDBY CDB200_STBY DESTINATION STANDBY CDB100_STBY ;
```

Step 9: Post Migration - Enable Services

Add any application services for the PDB to Cluster Ready Services (CRS), associating them with the PDB and correct database role in the destination CDB (CDB100), and remove the corresponding service from the source CDB (CDB200).

- For each service on both the primary and standby environments, run the following:

```
PRIMARY_HOST $ srvctl add service -db cdb100 -s SERVICE_NAME
 -pdb pdb001 -role [PRIMARY|PHYSICAL_STANDBY]….
STANDBY_HOST $ srvctl add service -db cdb100_stby -s SERVICE_NAME
 -pdb pdb001 -role [PRIMARY|PHYSICAL_STANDBY]….

<PRIMARY_HOST>PRIMARY_HOST $ srvctl remove service -db cdb200 -s
SERVICE_NAME
STANDBY_HOST $ srvctl remove service -db cdb200_stby -s SERVICE_NAME
```

- Start the required services for the appropriate database role.

1. Start each `PRIMARY` role database service.

```
PRIMARY_HOST $ srvctl start service -db cdb100 -s SERVICE_NAME
```

2. Start each `PHYSICAL_STANDBY` role database service.

```
STANDBY_HOST $ srvctl start service -db cdb100_stby -s SERVICE_NAME
```

PDB Failover Use Case

This is a very rare use case since a real disaster that encompasses CDB, cluster, or site failure should always leverage a complete CDB Data Guard failover operation to bound downtime, reduce potential data loss, and reduce administrative steps.

Even with widespread logical or data corruptions or inexplicable database hangs, it's more efficient to issue a CDB Data Guard role transition operation because the source environment may be suspect and root cause analysis may take a long time.

When does a PDB failover operation make sense? PDB failover may be viable if the application is getting fatal errors such as data integrity or corruption errors, or simply is not performing well (not due to system resources). If the source CDB and its corresponding PDBs

are still running well, and the standby did not receive any errors for the target sick PDB, then you can fail over just the target sick PDB from the standby without impacting any other PDBs in the source primary CDB.

The process below describes how to set up a PDB failover of a sick PDB that migrates the standby's healthy PDB from the source CDB standby (CDB100_STBY) to an empty destination CDB (CDB200). Before starting the migration, the destination CDB must be created and it must have the same database options as the source standby CDB. The destination CDB will contain no PDBs. The source and destination CDBs share the same storage and no data file movement is performed.

Prerequisites

Make sure your environment meets these prerequisites for the use case.

In addition to the prerequisites listed in the PDB switchover use case, above, the following prerequisites exist for failing over.

- Oracle recommends that you shut down the services on both the primary and the standby that are accessing the PDB before starting the migration process.

  If the PDB is not closed on the primary before running the `DGMGRL MIGRATE PLUGGABLE DATABASE` command, an error is returned stating that you will incur data loss. Closing the PDB on the primary resolves this issue. All existing connections to the PDB are terminated as part of the migration.

Assuming a destination CDB is already in place and patched correctly on the standby site, the entire process of moving the PDB can be completed in less than 15 minutes.

Additional Considerations

The following steps assume the source CDB database (either primary for migration or standby for failover) and the destination CDB database have access to the same storage, so copying data files is not required.

- Oracle Active Data Guard is required for the source CDB standby for failover operations.
- Create an empty CDB to be the destination for the PDB on the same cluster as the source CDB.
- Ensure that the TEMP file in the PDB has already been created in the source CDB standby before performing the migration.
- If the destination CDB is a later Oracle release the PDB will be plugged in but left closed to allow for manual upgrade as a post-migration task.
- After processing is completed, you may need to clean up leftover database files from the source databases.
- The plugin operation at the destination CDB is performed with `STANDBYS=NONE`, so you will need to manually enable recovery at any standby databases upon completion of the migration. See Making Use Deferred PDB Recovery and the STANDBYS=NONE Feature with Oracle Multitenant (Doc ID 1916648.1) for steps to enable recovery of a PDB.

Configuring PDB Failover

You configure the DR PDB failover use case in the following steps.

In this use case, the example topology has source primary CDB100 with 3 PDBs (PDB001, PDB002, PDB003). CDB100 also has a Data Guard physical standby (CDB100_STBY).

On the same environment as the standby CDB, we will create a new CDB (CDB200) which is a read-write database that becomes the new host for one of the source PDBs.

Step 1: Extract PDB Clusterware managed services on the source database

Determine any application and end user services created for the source PDB that have been added to CRS.

Because there are certain service attributes such as database role not stored in the database, the detail attributes should be retrieved from CRS using `SRVCTL CONFIG SERVICE`.

1. Retrieve the service names from the primary PDB (PDB002 in our example).

```
PRIMARY_HOST $ sqlplus sys@cdb100 as sysdba
SQL> alter session set container=pdb002;
SQL> select name from dba_services;
```

2. For each service name returned, retrieve the configuration including `DATABASE_ROLE`.

```
PRIMARY_HOST $ srvctl config service -db cdb100 -s SERVICE_NAME
```

Step 2: Create an empty target database

Create an empty CDB (CDB200 in our example) on the same cluster as the source standby CDB (CDB100_STBY) which will be the destination for the PDB (PDB002).

Allocate resources for this CDB to support the use of the PDB while it remains in this CDB.

Step 3: Create an Oracle Data Guard configuration for the empty target database

To allow Data Guard broker to access the new CDB (CDB200), it must be part of a Data Guard configuration. This configuration can consist of only a primary database.

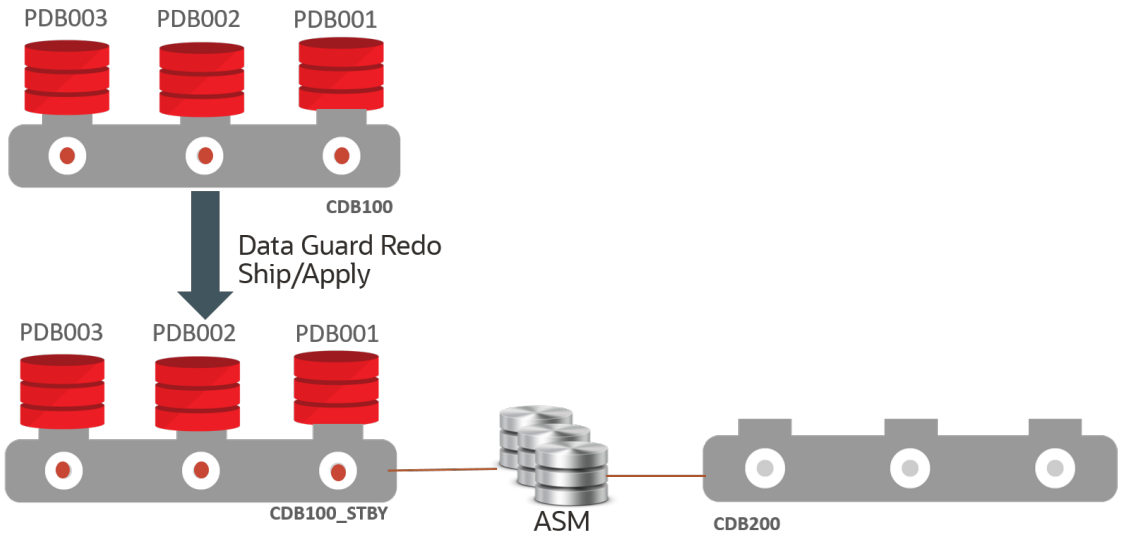
1. Configure the database for broker.

```
STANDBY_HOST $ sqlplus sys@cdb200 as sysdba
SQL> alter system set dg_broker_config_file1='+DATAC1/cdb200/
dg_broker_1.dat';
SQL> alter system set dg_broker_config_file2='+DATAC1/cdb200/
dg_broker_2.dat';
SQL> alter system set dg_broker_start=TRUE;
```

2. Create the configuration and add the database as the primary.

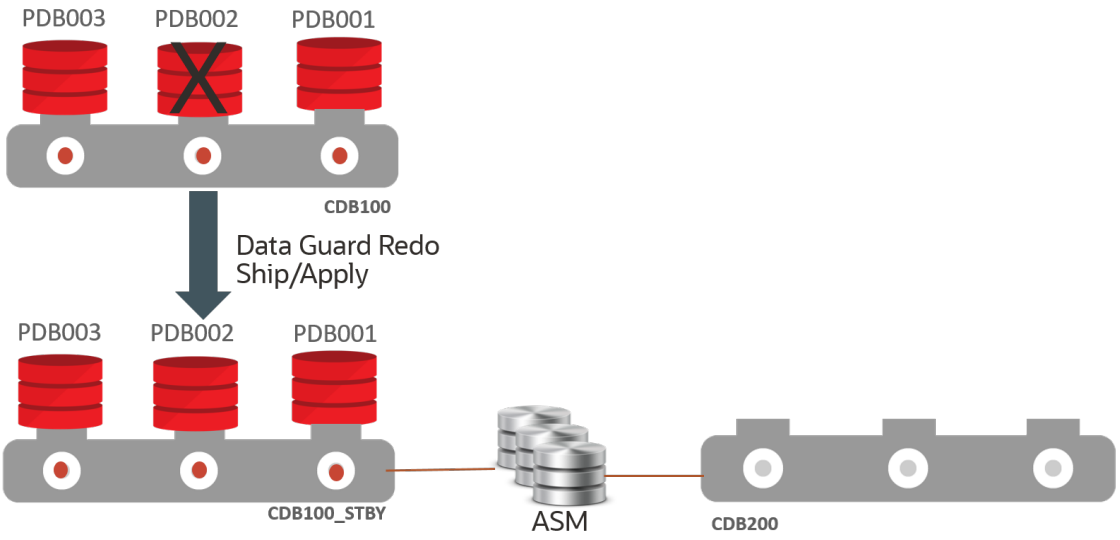
```
STANDBY_HOST $ dgmgrl
DGMGRL> connect sys@cdb200 as sysdba
DGMGRL> create configuration failover_dest as primary database is cdb200
 connect identifier is 'cdb200';
DGMGRL> enable configuration;
```

The configuration should resemble the image below.



In this image, the source primary CDB (CDB100) and all PDBs are running normally. the source standby CDB (CDB100_STBY) must run in Active Data Guard mode to allow for the "unplug" operation to succeed without impacting other PDBs. The destination CDB (CDB200) is currently empty.

Assume that one of the source primary PDBs (PDB002) experiences a failure, as shown in the image below, which requires a long recovery period, but the failure does not impact the other PDBs (PDB001 and PDB003), and the standby for the source CDB continues to apply redo without error.



This configuration will use files from PDB002 at the standby site (CDB100_STBY) to plug into the destination CDB (CDB200) to restore read/write application access and then drop the sick PDB (PDB002) from the source primary CDB (CDB100). This will not be a native unplug operation because native unplug requires a read/write CDB and in this scenario we're extracting from the standby.

Step 4: Stop services for the failed PDB

Although not required, stop all services on both the source primary database and any standby database(s) pertaining to the PDB (PDB002) to be migrated.

The following commands stop all services defined in CRS but does not close the PDB.

```
SOURCE_PRIMARY $ srvctl stop service -d CDB100 -pdb PDB002
SOURCE_PRIMARY $ srvctl stop service -d CDB100_STBY -pdb PDB002
```

Step 5: Fail over the PDB from the standby

Fail over the sick PDB (PDB002) from the standby CDB (CDB100_STBY) to the destination CDB (CDB200).

1. Start a `DGMGRL` session connecting to the source configuration standby database (CDB100_STBY).

   You must connect to the source standby database as SYSDBA using something similar to the following:

   ```
   $ dgmgrl
   DGMGRL> connect sys@cdb100_stby_inst1 as sysdba
   ```

2. Run the `DGMGRL MIGRATE PLUGGABLE DATABASE` command to perform the failover.

> **Note:**
>
> The `DGMGRL FAILOVER` command has a similar format to the `MIGRATE PLUGGABLE DATABASE` command.
> Do not use the `STANDBY FILES` keyword for the failover operation.
>
> If data loss is detected (SCN in the header of the first `SYSTEM` tablespace standby data file is less than the corresponding SCN of the file in the primary) and `IMMEDIATE` has not been specified, the `MIGRATE PLUGGABLE DATABASE` command will fail. The most common reason is that the PDB in the primary CDB is still open, the PDB on the primary should be closed before attempting a failover.
>
> You must resolve the SCN discrepancy or accept the data loss with the `IMMEDIATE` clause.

3. Fail over the PDB

   See Full Example Commands with Output for examples with complete output.

   The CONNECT alias should connect to the destination primary instance that is on the same host as the broker CLI session/source standby database instance to ensure that the plugin operation can access the PDB unplug manifest file that will be created.

   > **Note:**
   >
   > In the following examples, you will be prompted for the SYSDBA password for the destination CDB (CDB200) when the broker attempts to connect to the CDB200_INST1 instance.

   - For non-TDE enabled environments:

     ```
     DGMGRL> migrate pluggable database PDB002 to container CDB200
      using '/tmp/PDB002.xml>' connect as sys@"CDB200_INST1";
     ```

   - For TDE enabled environments:

     ```
     DGMGRL> migrate pluggable database PDB002 to container CDB200
      using '/tmp/PDB002.xml>' connect as sys@"CDB200_INST1"
      secret "some_value"
      keystore identified by "destination_keystore_password"
      keyfile '/tmp/pdb002_key.dat'
      source keystore identified by "source_keystore_password";
     ```

     > **Note:**
     >
     > For TDE environments, if `SECRET`, `KEYSTORE`, `KEYFILE`, or `SOURCE KEYSTORE` are not specified in the command line, the `MIGRATE PLUGGABLE DATABASE` command fails.

Once the connection to the destination is established the command will:

1. Perform all necessary validations for the failover operation

2. If TDE is enabled, export the TDE keys for the PDB from the source standby keystore
3. Stop redo apply on the source standby if it is running
4. Create the manifest on the standby at the location specified in the command using the `DBMS_PDB.DESCRIBE` command
5. Disable recovery of the PDB at the source standby
6. If TDE is enabled, import TDE keys into the destination CDB keystore to allow the plugin to succeed
7. Plugin the PDB in the destination database using the standby's data files (`NOCOPY` clause) and with `STANDBYS=NONE`.
8. Open the PDB in all instances of the destination primary database
9. If TDE is enabled, issue `ADMINISTER KEY MANAGEMENT USE KEY` in the context of the PDB to associate the imported key and the PDB.
10. Unplug the PDB from the source primary. If errors occur on unplug messaging is provided to user to perform cleanup manually
11. If unplug succeeds, drop the PDB from the source primary with the `KEEP DATAFILES` clause. This will also drop the PDB in all of the source standby databases.

Step 6: Post Migration - Enable Services

Add any application services for the PDB to Cluster Ready Services (CRS), associating them with the PDB and correct database role in the destination CDB, and remove the corresponding service from the source CDB.

1. For each service on both the primary and standby environments, run the following:

```
DESTINATION_PRIMARY_HOST $ srvctl add service -db cdb200 -s SERVICE_NAME
 -pdb pdb002 -role [PRIMARY|PHYSICAL_STANDBY]….

SOURCE_PRIMARY_HOST $ srvctl remove service -db cdb100 -s SERVICE_NAME
SOURCE_STANDBY_HOST $ srvctl remove service -db cdb100_stby -s SERVICE_NAME
```

2. Start the required services for the appropriate database role.

   Start each `PRIMARY` role database service

```
DESTINATION_PRIMARY_HOST $ srvctl start service -db cdb200 -s SERVICE_NAME
```

Step 7: Back up the PDB

Back up the PDB in the destination CDB (CDB200) to allow for recovery going forward.

```
DESTINATION_PRIMARY_HOST $ rman
RMAN> connect target sys@cdb200
RMAN> backup pluggable database pdb002;
```

Step 8: Optionally enable recovery of the PDB

Follow the steps in Making Use Deferred PDB Recovery and the STANDBYS=NONE Feature with Oracle Multitenant (Doc ID 1916648.1) to enable recovery of the PDB at any standby databases to establish availability and disaster recovery requirements.

Step 9: Optionally to Migrate Back

See the migration steps in Configuring PDB Switchover.

Resolving Errors

For cases where the plugin to the destination primary CDB succeeds but there are issues such as file not found at the destination standby, you can use the GRP created on the destination CDB standby database to help in resolution.

If the broker detects an error at the standby it ends execution without removing the GRP, it can be used to help resolve errors. The GRP name is displayed in the output from the CLI command execution.

Before using this method, ensure that all patches from the prerequisites section have been applied.

1. Turn off redo apply in Data Guard Broker so it does not automatically start

   ```
   DGMGRL> edit database CDB200_STBY set state='APPLY-OFF';
   ```

2. Restart the destination CDB standby in mount mode, ensuring in RAC environments only one instance is running.

   - For Oracle RAC

     ```
     $ srvctl stop database -d cdb200_stby -o immediate

     $ srvctl start instance -d cdb200_stby -i cdb200s1 -o mount
     ```

   - For SIDB

     ```
     SQL> shutdown immediate

     SQL> startup mount
     ```

3. Connect to the PDB in the destination CDB standby database and disable recovery of the PDB.

   ```
   SQL> alter session set container=pdb001;

   SQL> alter pluggable database disable recovery;
   ```

4. Connect to the CDB$root of the destination CDB standby database and flashback the standby database.

   ```
   SQL> alter session set container=cdb$root;

   SQL> flashback database to restore point <GRP from execution>;
   ```

5. Repair any issues that caused redo apply to fail (e.g. missing ASM aliases).

6. Staying in mount mode on the CDB standby, start redo apply.

   ```
   SQL> recover managed standby database disconnect;
   ```

   Redo apply will now start applying all redo from the GRP forward, including rescanning for all the files for the newly plugged in PDB. The flashback GRP rolls back the destination CDB standby to the point where the PDB is unknown to the standby, so the disabling of recovery for the PDB is backed out as well.

Steps 1-6 can be repeated as many times as is required until all files are added to the standby and additional redo is being applied at which point you would:

1. Stop recovery

   ```
   DGMGRL> edit database CDB200_STBY set state='APPLY-OFF';
   ```

2. Connect to the CDB$root of the destination CDB standby database and drop the GRP from the destination standby database:

```
SQL> drop restore point <GRP from execution>;
```

3. Restart redo apply

```
DGMGRL> edit database CDB200_STBY set state='APPLY-ON';
```

If you continue to have issues and require that your CDB standby database maintain protection of additional PDBs in the standby during problem resolution:

- Disable recovery of the PDB as noted above
- Restart redo apply so that the other PDBs in the CDB standby are protected
- Follow the Enable Recovery steps in Making Use Deferred PDB Recovery and the STANDBYS=NONE Feature with Oracle Multitenant (Doc ID 1916648.1) to enable recovery of the failed PDB.
- Drop the GRP from the destination CDB standby.

During testing if there are repetitive errors on the standby that cannot be resolved:

1. Enable PDB operation debugging for redo apply on the standby.

```
SQL> alter system set "_pluggable_database_debug"=256 comment='set to help
debug PDB plugin issues for PDB100, reset when done' scope=both;
```

2. Follow the steps above to flashback the destination CDB standby database.
3. Restart redo apply.

After the new failure, gather the redo apply trace files from the standby host that was running redo apply (..../trace/<SID>_pr*.trc) and open a bug.

Once debugging is done:

1. Reset the parameter to turn off debugging.

```
SQL> alter system reset "_pluggable_database_debug" scope=spfile;
```

2. Bounce the CDB standby database.

Reference

Note that the following examples may generate different output as part of the `DGMGRL MIGRATE` command than you will see while executing the command, based on the different states of PDBs and items found by `DGMGRL` running prechecks in your environment. In addition, Oracle does not ship message files with bug fixes, so instead of displaying full messages you may receive something similar to the following:

```
Message 17241 not found; product=rdbms; facility=DGM
```

This does not mean it's an error or a problem, it means that the text we want to display is missing from the message file. All messages are displayed in their entirety in the first release containing all of the fixes.

Full Example Commands with Output

The following are examples of the commands with output.

Example 35-1 Migrate without TDE

```
DGMGRL> MIGRATE PLUGGABLE DATABASE PDB001 TO CONTAINER CDB200
 USING '/tmp/PDB001.xml' CONNECT AS sys/password@cdb200_inst1
```

```
 STANDBY  FILES sys/standby_asm_sys_passwd@standby_asm_inst1
 SOURCE STANDBY CDB100_STBY DESTINATION STANDBY CDB200_STBY ;

Beginning migration of pluggable database PDB001.
Source multitenant container database is CDB100.
Destination multitenant container database is CDB200.
Connecting to "+ASM1".
Connected as SYSASM.
Stopping Redo Apply services on multitenant container database cdb200_stby.
The guaranteed restore point "<GRP name>" was created for multitenant
container database "cdb2001_stby".
Restarting redo apply services on multitenant container database cdb200_stby.
Closing pluggable database PDB001 on all instances of multitenant container
database CDB100.
Unplugging pluggable database PDB001 from multitenant container database
cdb100.
Pluggable database description will be written to /tmp/pdb001.xml
Dropping pluggable database PDB001 from multitenant container database CDB100.
Waiting for the pluggable database PDB001 to be dropped from standby
multitenant container database cdb100_stby.
Creating pluggable database PDB100 on multitenant container database CDB200.
Checking whether standby multitenant container database cdb200_stby has added
all data files for pluggable database PDB001.
Opening pluggable database PDB001 on all instances of multitenant container
database CDB200.
The guaranteed restore point "<GRP_name>" was dropped for multitenant
container database "cdb200_stby".
Migration of pluggable database PDB001 completed.

Succeeded.
```

Example 35-2 Migrate with TDE

```
DGMGRL> MIGRATE PLUGGABLE DATABASE PDB001 TO CONTAINER CDB200 USING '/tmp/
pdb001.xml'
 CONNECT AS sys/password@cdb200_inst1 SECRET "some_value"
 KEYSTORE IDENTIFIED BY "destination_TDE_keystore_passwd"
 STANDBY FILES sys/standby_ASM_sys_passwd@standby_asm_inst1
 SOURCE STANDBY cdb100_stby DESTINATION STANDBY cdb200_stby;

Master keys of the pluggable database PDB001 to need to be migrated.
Keystore of pluggable database PDB001 is open.
Beginning migration of pluggable database PDB001.
Source multitenant container database is cdb100.
Destination multitenant container database is cdb200.
Connecting to "+ASM1".
Connected as SYSASM.
Stopping Redo Apply services on multitenant container database cdb200_stby.
The guaranteed restore point "..." was created for multitenant container
database "cdb200_stby".
Restarting redo apply services on multitenant container database cdb200_stby.
Closing pluggable database PDB001 on all instances of multitenant container
database cdb100.
Unplugging pluggable database PDB001 from multitenant container database
cdb100.
```

```
Pluggable database description will be written to /tmp/pdb001.xml
Dropping pluggable database PDBT001 from multitenant container database
cdb100.
Waiting for the pluggable database PDB001 to be dropped from standby
multitenant container
database cdb100_stby.
Creating pluggable database PDB1001 on multitenant container database cdb200.
Checking whether standby multitenant container database cdb200_stby has added
all data files for pluggable database PDB001.
Stopping Redo Apply services on multitenant container database cdb200_stby.
Opening pluggable database PDB001 on all instances of multitenant container
database cdb400.
The guaranteed restore point "..." was dropped for multitenant container
database "cdb200_stby".

Please complete the following steps to finish the operation:
1. Copy keystore located in <cdb200 primary keystore location> for migration
destination primary database to <cdb200 standby keystore location> for
migration destination standby database.
2. Start DGMGRL, connect to multitenant container database cdb200_stby, and
issue command "EDIT DATABASE cdb200_stby SET STATE=APPLY-ON".
3. If the clusterware is configured on multitenant container databases cdb200
or cdb200_stby, add all non-default services for the migrated pluggable
database in cluster ready services.
Migration of pluggable database PDB001 completed.

Succeeded.
```

Example 35-3 Failover without TDE

```
DGMGRL> migrate pluggable database PDB002 immediate to container CDB200
 using '/tmp/<pdb002.xml>';
Username: USERNAME@cdb200
Password:
Connected to "cdb200"
Connected as SYSDBA.

Beginning migration of pluggable database pdb002.
Source multitenant container database is cdb100_stby.
Destination multitenant container database is cdb200.

Connected to "cdb100"
Closing pluggable database pdb002 on all instances of multitenant container
database cdb100.
Continuing with migration of pluggable database pdb002 to multitenant
container database cdb200.
Stopping Redo Apply services on source multitenant container database
cdb100_stby.
Succeeded.
Pluggable database description will be written to /tmp/pdb002.xml.
Closing pluggable database pdb002 on all instances of multitenant container
database cdb100_stby.
Disabling media recovery for pluggable database pdb002.
Restarting redo apply services on source multitenant container database
cdb100_stby.
```

```
Succeeded.
Creating pluggable database pdb002 on multitenant container database cdb200.
Opening pluggable database pdb002 on all instances of multitenant container
database cdb200.
Unplugging pluggable database pdb002 from multitenant container database
cdb100.
Pluggable database description will be written to /tmp/pdb002_temp.xml.
Dropping pluggable database pdb002 from multitenant container database cdb100.
Unresolved plug in violations found while migrating pluggable database pdb002
to multitenant container database cdb200.
Please examine the PDB_PLUG_IN_VIOLATIONS view to see the violations that
need to be resolved.
Migration of pluggable database pdb002 completed.
Succeeded.
```

Example 35-4 Filover with TDE

NOTE: ORA-46655 errors in the output can be ignored.

```
DGMGRL> migrate pluggable database PDB002 to container CDB200
 using '/tmp/PDB002.xml>' connect as sys@"CDB200" secret "some_value"
 keystore identified by "destination_keystore_password" keyfile '/tmp/
pdb002_key.dat'
 source keystore identified by "source_keystore_password";
Connected to "cdb200"
Connected as SYSDBA.
Master keys of the pluggable database PDB002 need to be migrated.
Keystore of pluggable database PDB002 is open.

Beginning migration of pluggable database PDB002.
Source multitenant container database is adg.
Destination multitenant container database is cdb200.

Connected to "cdb1001"
Exporting master keys of pluggable database PDB002.
Continuing with migration of pluggable database PDB002 to multitenant
container database cdb200.
Stopping Redo Apply services on multitenant container database adg.
Pluggable database description will be written to /tmp/PDB002.xml.
Closing pluggable database PDB002 on all instances of multitenant container
database adg.
Disabling media recovery for pluggable database PDB002.
Restarting redo apply services on multitenant container database adg.
Unplugging pluggable database PDB002 from multitenant container database
cdb100.
Pluggable database description will be written to /tmp/ora_tfilSxnmva.xml.
Dropping pluggable database PDB002 from multitenant container database cdb100.
Importing master keys of pluggable database PDB002 to multitenant container
database cdb200.
Creating pluggable database PDB002 on multitenant container database cdb200.
Opening pluggable database PDB002 on all instances of multitenant container
database cdb200.
ORA-46655: no valid keys in the file from which keys are to be imported

Closing pluggable database PDB002 on all instances of multitenant container
database cdb200.
```

```
Opening pluggable database PDB002 on all instances of multitenant container
database cdb200.

Please complete the following steps to finish the operation:
If the Oracle Clusterware is configured on multitenant container database
CDB200, add all non-default services for the migted pluggable database in
Cluster Ready Services.

Migration of pluggable database PDB002 completed.
Succeeded.
```

Keyword Definitions

The `DGMGRL MIGRATE` command keywords are explained below.

Syntax

```
DGMGRL> MIGRATE PLUGGABLE DATABASE pdb-name
TO CONTAINER dest-cdb-name
USING XML-description-file
CONNECT AS { /@dest-cdb-connect-identifer |
 dest-cdb-user/dest-cdb-password@dest-cdb-connect-identifier}
[SECRET "secret" KEYSTORE IDENTIFIED BY ( EXTERNAL STORE | wallet-password) ;]
STANDBY FILES { /@asm-instance-connect-identifer |
 sysasm-user/sysasm-password@asm-instance-connect-identifier}
SOURCE STANDBY source-standby-cdb-name
DESTINATION STANDBY dest-standby-cdb-name
[TIMEOUT timeout]
```

These are the keyword definitions used on the PDB migrate command

- *pdb-name* - The name of the PDB to be migrated.
- *dest-cdb-name* - The database unique name of the CDB to receive the PDB to be migrated.
- *XML-description-file* - An XML file that contains the description of the PDB to be migrated. This file is automatically created by the SQL statements executed by the `MIGRATE PLUGGABLE DATABASE` command and the location of the file must be directly accessible by both the source and destination primary database instances. It cannot exist prior to command execution.
- *dest-cdb-user* - The user name of the user that has `SYSDBA` access to the destination CDB.
- *dest-cdb-password* - The password associated with the user name specified for *dest-cdb-user*.
- *dest-cdb-connect-identifier* - An Oracle Net connect identifier used to reach the destination CDB.
- *secret* - A word used to encrypt the export file containing the exported encryption keys of the source PDB. This clause is only required for TDE enabled environments.
- *keyfile* - A data file that contains the exported encryption keys for the source PDB. This file is created by SQL statements executed by the `MIGRATE PLUGGABLE DATABASE` command in the failover use case and the location of the file must be directly accessible by the source standby instance and the destination primary instance.

- *wallet-password* - The password of the destination CDB keystore containing the encryption keys. This is required if the source PDB was encrypted using a password keystore in TDE enabled environments.
- *asm-instance-connect-identifier* - The connect identifier to the ASM instance having the source standby database file.
- *sysasm-user* - A user having SYSASM privilege for ASM instance.
- *sysasm-password* - The password for *sysasm-user*.
- *source-standby-cdb-name* - DB_UNIQUE_NAME of the migration source CDB's standby database.
- *dest-standby-cdb-name* - DB_UNIQUE_NAME of the migration destination CDB's standby database.
- *timeout* - The timeout value in seconds when waiting for the destination standby database picks up the data files during migration. This is optional. The default if the TIMEOUT clause is omitted is 5 minutes.

Messages

The following is the list of messages possibly produced by the DGMGRL MIGRATE function:

For generic processing

17180 - "Pluggable database %s must be open prior to starting a migration operation."

17217 - "Migration cannot be performed when the source multitenant container database (%(1)s) is a physical standby running a different version of Oracle than %(2)s."

17235 - "Investigate why the pluggable database %s could not be unplugged."

17236 - "Resolve the issue and then manually unplug and drop the pluggable database from database %s."

17237 - "Migration of pluggable database %s completed."

17238 - "Migration of pluggable database %s completed with warnings."

17239 - "Failed to migrate pluggable database %s."

17240 - "Media recovery is disabled for pluggable database %(1)s on multitenant container database %(2)s."

17241 - "Warning: either source or destination multitenant container database does not have local undo enabled."

17242 - "Migration from pluggable database %s not possible since it is either a snapshot child or snapshot parent."

17243 - "Pluggable database %s could not be opened because it was migrated to a database running a higher Oracle version."

17244 - "Please run the appropriate upgrade procedures prior to opening the pluggable database."

17245 - "The file location specified (%s) is not accessible."

17246 - "A file name was not specified."

17247 - "An invalid file name (%s) was specified."

17248 - "Retry the command after the lag is resolved or use the IMMEDIATE option to ignore the data loss."

17249 - "Media recovery is disabled for pluggable database %(1)s on multitenant container database %(2)s."

17250 - "Warning: either source or destination multitenant container database does not have local undo enabled."

17251 - "Migration from pluggable database %s not possible since it is either a snapshot child or snapshot parent."

For addition of TDE support

17413 - "Failed to open keystore of pluggable database %s."

17414 - "Keystore of pluggable database %s is not open."

17415 - "Keystore password of pluggable database %s is required."

17416 - "Keystore password of multitenant container database %s is required."

17427 - "Unable to fetch keystore status of pluggable database %s."

17428 - "Keystore of pluggable database %s is open."

17429 - "Keystore password of pluggable database %s is not correct."

17430 - "Keystore password of multitenant container database %s is not correct."

For standby file support

17510 - "The standby database \"%s\" is not using or connected to an ASM instance."

17511 - "The standby database of the source multitenant container database is using a different ASM disk group than that of the destination multitenant container database."

17512 - "The initialization parameter DB_FILE_NAME_CONVERT of the migration destination standby database is not NULL."

17513 - "The initialization parameter STANDBY_FILE_MANAGEMENT of the migration source standby database is not AUTO."

17514 - "The multitenant container database %s is not a physical standby database."

17515 - "The ASM alias of data file %s is not in the expected location."

17516 - "A multitenant container standby database in the Data Guard Broker configuration must be specified."

17517 - "Data files cannot be reused when the source multitenant container database is a standby database."

17518 - "The ASM alias %s refers to an ASM file that is not in the expected location."

17519 - "The guaranteed restore point \"%(1)s\" was created for multitenant container database \"%(2)s\"."

17520 - "The guaranteed restore point \"%(1)s\" was dropped for multitenant container database \"%(2)s\"."

17521 - “Connected as SYSASM.”

17522 - "The multitenant container database \"%(1)s\" failed to find the data file \"%(2)s\"."

17523 - "The multitenant container database \"%s\" is in an unstable state."

17524 - "The multitenant container database \"%(1)s\" can be restored using the restore point \"%(2)s\"."

17525 - "Redo apply stopped or failed on multitenant container database \"%s\"."

17530 - "The standby multitenant container database %(1)s failed to add all data files for pluggable database %(2)s."

17532 - "Failed to drop the pluggable database %(1)s from standby multitenant container database %(2)s."

17533 - "The specified file (%s) must not exist."

17534 - "A path was not specified."

17536 - "Unable to fetch keystore mode of pluggable database %s."

17537 - "KEYFILE and SOURCE IDENTIFIED BY clauses are required."

17539 - "Importing master keys of pluggable database %(1)s to multitenant container database %(2)s."

Sample Oracle Database Net Services Connect Aliases

The following Net Services connect aliases must be accessible to DGMGRL when starting the broker session. This can be through default tnsnames.ora location or by setting TNS_ADMIN in the environment before starting DGMGRL.

PDB Switchover

The host names in the following examples reference Oracle Single Client Access Name (SCAN) host names. There is overlap in the host names between the source and destination databases as they must reside on the same hosts. In all cases the connect strings should connect to the cdb$root of the database.

Source primary database

```
CDB100 =
          (DESCRIPTION =
            (CONNECT_TIMEOUT=120)(TRANSPORT_CONNECT_TIMEOUT=90)(RETRY_COUNT=3)
            (ADDRESS =
                (PROTOCOL = TCP)
                (HOST = <source-primary-scan-name>)
                (PORT = <source-primary-listener-port>)
            )
            (CONNECT_DATA =
              (SERVER = DEDICATED)
              (SERVICE_NAME = <source-primary-service-name>)
              (FAILOVER_MODE =
                  (TYPE = select)
                  (METHOD = basic)
              )
             )
          )
```

Source primary database local instance

```
CDB100_INST1 =
          (DESCRIPTION =
            (CONNECT_TIMEOUT=120)(TRANSPORT_CONNECT_TIMEOUT=90)(RETRY_COUNT=3)
            (ADDRESS =
                (PROTOCOL = TCP)
                (HOST = <source-primary-scan-name>)
                (PORT = <source-primary-listener-port>)
            )
            (CONNECT_DATA =
              (SERVER = DEDICATED)
              (SERVICE_NAME = <source-primary-cdb$root-service-name>)
              (INSTANCE_NAME = <source-primary-local-instance-name>)
            )
          )
```

Destination primary database

```
CDB200=
    (DESCRIPTION=
      (CONNECT_TIMEOUT=120)(TRANSPORT_CONNECT_TIMEOUT=90)(RETRY_COUNT=3)
      (ADDRESS=
        (PROTOCOL= TCP)
        (HOST= <source-primary-scan-name>)
        (PORT= <source-primary-listener-port>))
      (CONNECT_DATA=
        (SERVER= DEDICATED)
        (SERVICE_NAME= <destination-primary-cdb$root-service-name>)))
```

Destination primary local instance

This must connect to an instance on the same host that dgmgrl is being executed

```
CDB200_INST1=
    (DESCRIPTION=
      (CONNECT_TIMEOUT=120)(TRANSPORT_CONNECT_TIMEOUT=90)(RETRY_COUNT=3)
      (ADDRESS=
        (PROTOCOL= TCP)
        (HOST= <source-primary-scan-name>)
        (PORT= <source-primary-listener-port>))
      (CONNECT_DATA=
        (SERVER= DEDICATED)
        (SERVICE_NAME= <destination-primary-cdb$root-service-name>)
        (INSTANCE_NAME = <destination-primary-local-instance-name>)
      )
    )
```

Source standby database

```
CDB100_STBY =
          (DESCRIPTION =
            (CONNECT_TIMEOUT=120)(TRANSPORT_CONNECT_TIMEOUT=90)(RETRY_COUNT=3)
            (ADDRESS =
```

```
                (PROTOCOL = TCP)
                (HOST = <source-standby-scan-name)
                (PORT = <source-standby-listener-port>)
              )
              (CONNECT_DATA =
                (SERVER = DEDICATED)
                (SERVICE_NAME = <source-standby-cdb$root-service-name>)
                (FAILOVER_MODE =
                    (TYPE = select)
                    (METHOD = basic)
                )
               )
            )
```

Destination standby database

```
CDB200_STBY =
           (DESCRIPTION =
             (CONNECT_TIMEOUT=120)(TRANSPORT_CONNECT_TIMEOUT=90)(RETRY_COUNT=3)
             (ADDRESS =
                 (PROTOCOL = TCP)
                 (HOST = <source-standby-scan-name>)
                 (PORT = <source-standby-listener-port>)
             )
             (CONNECT_DATA =
               (SERVER = DEDICATED)
               (SERVICE_NAME = <destination-standby=cdb$root-service-name>)
               (FAILOVER_MODE =
                   (TYPE = select)
                   (METHOD = basic)
               )
              )
           )
```

Standby environment ASM

This must connect to an ASM instance running on the same host as one instance each of the source standby and destination standby

```
STANDBY_ASM_INST1=
    (DESCRIPTION=
      (CONNECT_TIMEOUT=120)(TRANSPORT_CONNECT_TIMEOUT=90)(RETRY_COUNT=3)
      (ADDRESS=
        (PROTOCOL= TCP)
        (HOST = <source-standby-scan-name>)
        (PORT= <source-standby-listener-port>))
      (CONNECT_DATA=
        (SERVER= DEDICATED)
        (SERVICE_NAME= +ASM)
        (INSTANCE_NAME=<ASM_instance_name>)
      )
    )
```

PDB Failover

Source primary database

```
CDB100 =
          (DESCRIPTION =
            (CONNECT_TIMEOUT=120)(TRANSPORT_CONNECT_TIMEOUT=90)(RETRY_COUNT=3)
            (ADDRESS =
                (PROTOCOL = TCP)
                (HOST = <source-primary-scan-name>)
                (PORT = <source-primary-listener-port>)
            )
            (CONNECT_DATA =
              (SERVER = DEDICATED)
              (SERVICE_NAME = <source-primary-cdb$root-service-name>)
              (FAILOVER_MODE =
                  (TYPE = select)
                  (METHOD = basic)
               )
             )
          )
```

Source standby database

```
CDB100_STBY =
          (DESCRIPTION =
            (CONNECT_TIMEOUT=120)(TRANSPORT_CONNECT_TIMEOUT=90)(RETRY_COUNT=3)
            (ADDRESS =
                (PROTOCOL = TCP)
                (HOST = <source-standby-scan-name>)
                (PORT = <source-standby-listener-port>)
            )
            (CONNECT_DATA =
              (SERVER = DEDICATED)
              (SERVICE_NAME = <source-standby-cdb$root-service-name>)
              (FAILOVER_MODE =
                  (TYPE = select)
                  (METHOD = basic)
               )
             )
          )
```

Source standby database local instance

This must connect to an instance on the same host that dgmgrl is being executed

```
CDB100_STBY_INST1=
    (DESCRIPTION=
      (CONNECT_TIMEOUT=120)(TRANSPORT_CONNECT_TIMEOUT=90)(RETRY_COUNT=3)
      (ADDRESS=
        (PROTOCOL= TCP)
        (HOST= <source-standby-scan-name>)
```

```
        (PORT= <source-standby-listener-port>))
      (CONNECT_DATA=
        (SERVER= DEDICATED)
        (SERVICE_NAME= <source-standby-cdb$root-service-name>)
        (INSTANCE_NAME = <source-standby-local-instance-name>)
      )
    )
```

Destination primary database

```
CDB200=
    (DESCRIPTION=
      (ADDRESS=
        (PROTOCOL= TCP)
        (HOST= <source-standby-scan-name>)
        (PORT= <source-standby-listener-port>))
      (CONNECT_DATA=
        (SERVER= DEDICATED)
        (SERVICE_NAME= <destination-primary-cdb$root-service-name>)))
```

Destination primary local instance

This must connect to an instance on the same host that dgmgrl is being executed

```
CDB200_INST1=
    (DESCRIPTION=
      (ADDRESS=
        (PROTOCOL= TCP)
        (HOST= <source-standby-scan-name>)
        (PORT= <source-standby-listener-port>))
      (CONNECT_DATA=
        (SERVER= DEDICATED)
        (SERVICE_NAME= <destination-primary-cdb$root-service-name>)
        (INSTANCE_NAME = <destination-primary-local-instance-name>)
      )
    )
```

Part IX

Full Site Switch in Oracle Cloud or On-Premises

36

Full Site Switch in Oracle Cloud or On-Premise

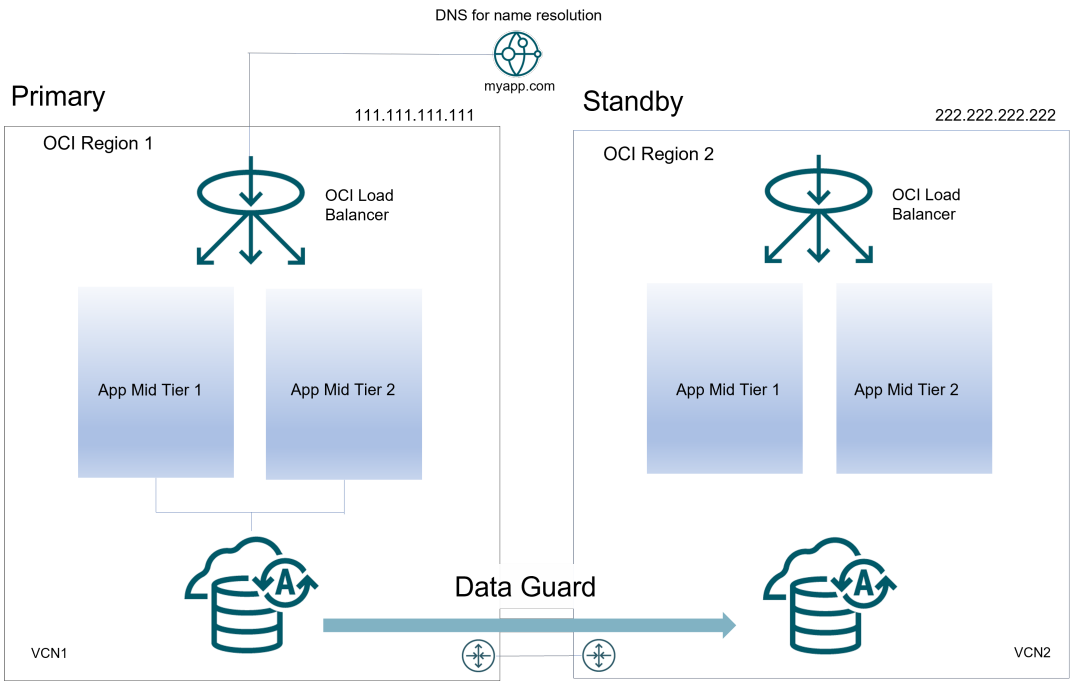
A complete-site or full site failure results in both the application and database tiers being unavailable. To maintain availability users must be redirected to a secondary site that hosts a redundant application tier and a synchronized copy of the production database. MAA best practice is to use Data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

During normal runtime operations the following occurs:

1. Client requests enter the client tier of the primary site and travel by the WAN traffic manager.
2. Client requests are sent to the application server tier.
3. Requests are forwarded through the active load balancer to the application servers.
4. Requests are sent into the database server tier.
5. The application requests, if required, are routed to an Oracle RAC instance.
6. Responses are sent back to the application and clients by a similar path.

The following illustrates the possible network routes before site switchover:

Figure 36-1 Sites before switchover

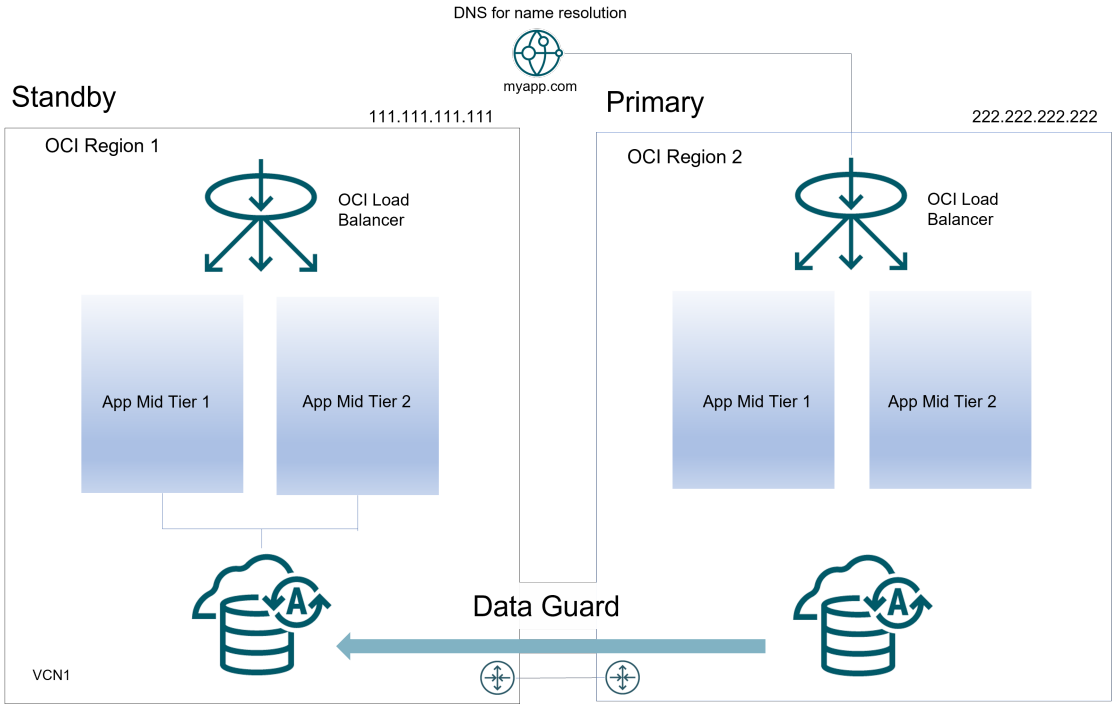


The following steps describe the effect of a site switchover:

1. The administrator has failed over or switched over the primary database to the secondary site. This is automatic if you are using Data Guard Fast-Start Failover. Autonomous Database on Dedicated Hardware supports Data Guard Fast-Start Failover.
2. The administrator starts the middle-tier application servers on the secondary site, if they are not running. In some cases the same middle-tier application servers can be leveraged if they do not reside in the failed site.
3. The wide-area traffic manager selection of the secondary site can be automatic for an entire site failure.
4. The wide-area traffic manager at the secondary site returns the virtual IP address of a load balancer at the secondary site and clients are directed automatically on the subsequent reconnect. In this scenario, the site failover is accomplished by an automatic domain name system (DNS) failover.

The following figure illustrates the network routes after site failover. Client or application requests enter the secondary site at the client tier and follow the same path on the secondary site that they followed on the primary site.

Figure 36-2 Sites after switchover



Failover also depends on the client's web browser. Most browser applications cache the DNS entry for a period. Consequently, sessions in progress during an outage might not fail over until the cache timeout expires. To resume service to such clients, close the browser and restart it.

Performing Role Transitions Between Regions

Examples below leverage Oracle Public Cloud. However similar steps can be done on-premise or hybrid cloud scenarios.

Failover to Another Region

A failover operation is performed when the primary site becomes unavailable, and it is commonly an unplanned operation. You can role-transition a standby database to a primary database when the original primary database fails and there is no possibility of recovering the primary database in a timely manner. There may or may not be data loss depending upon whether your primary and target standby databases were consistent at the time of the primary database failure.

To perform a manual failover in a DR configuration follow these steps:

1. Switchover DNS name.

   Perform the required DNS push in the DNS server hosting the names used by the system or alter the file host resolution in clients to point the front-end address of the system to the public IP used by load balancer in site2. For scenarios where DNS is used for the external front-end resolution (OCI DNS, commercial DNS, etc.), appropriate API can be used to push the change. An example that push this change in an OCI DNS:

   The following is an OCI client script that updates a front end DNS entry, such as ordscsdroci.domainexample.com, to the site 1 load balancer's public IP address (for example: 111.111.111.123).

   ```
   oci dns record rrset update
    --config-file /home/opc/scripts/.oci_ordscsdr/config
    --zone-name-or-id "domainexample.com"
    --domain "ordscsdroci.domainexample.com"
    --rtype "A"
    --items
   '[{"domain":"ordscsdroci.domainexample.com","rdata":"111.111.111.123","rtyp
   e":"A","ttl":60}]'
    --force
   ```

2. Failover database.

   On Oracle Cloud:

   Use Oracle Control Plane and issue a Data Guard switchover or failover operation.

   On-Premises:

   Use Data Guard broker in secondary database host to perform the failover. As user oracle:

   ```
   [oracle@drdbwlmp1b ~]$ dgmgrl sys/your_sys_password@secondary_db_unqname
   DGMGRL> failover to "secondary_db_unqname"
   ```

3. Start the servers in the secondary site.

   Restart the secondary application servers.

Switchover

A switchover is a planned operation where an administrator reverts the roles of the two sites. The roles change from the primary to the standby as well as from standby to primary. This is known as a manual switchover. To perform a manual switchover follow these steps:

1. Propagate any pending configuration changes.

   For non-database files, you can use rsync or Object Storage Service (OSS) to replicate to your secondary site.

2. Stop servers in the primary site.

Use scripts to stop managed servers / mid tiers in primary Site.

3. Switchover DNS name

   Perform the required DNS push in the DNS server hosting the names used by the system or alter the file host resolution in clients to point the front-end address of the system to the public IP used by load balancer in site 2. For scenarios where DNS is used for the external front-end resolution (OCI DNS, commercial DNS, etc.), appropriate API can be used to push the change.

   The following example pushes this change in an OCI DNS.

   The OCI client script updates the front end DNS entry, for example ordscsdroci.domainexample.com, to the site1 load balancer's public IP address (for example: 111.111.111.123).

   ```
   oci dns record rrset update
    --config-file /home/opc/scripts/.oci_ordscsdr/config
    --zone-name-or-id "domainexample.com"
    --domain "ordscsdroci.domainexample.com"
    --rtype "A"
    --items
   '[{"domain":"ordscsdroci.domainexample.com","rdata":"111.111.111.123","rtyp
   e":"A","ttl":60}]'
    --force
   ```

   Note that the TTL value of the DNS entry will affect to the effective RTO of the switchover: if the TTL is high (example, 20 mins), the DNS change will take that time to be effective in the clients. Using lower TTL values will make this to be faster, however, this can cause an overhead because the clients check the DNS more frequently. A good approach is to set the TTL to a low value temporarily (example, 1 min), before the change in the DNS. Then, perform the change, and once the switchover procedure is completed, set the TTL to the normal value again.

4. Perform database switchover.

   On Oracle Cloud:

   Use Oracle Control Plane and issue a Data Guard switchover operation.

   On-Premises:

   Use Data Guard broker on the primary database host to perform the switchover.

   As user oracle:

   ```
   $ dgmgrl sys/your_sys_password@primary_db_unqname
   DGMGRL> switchover to "secondary_db_unqname"
   ```

5. Start the servers in secondary site (new primary).

   Restart the secondary managed servers and mid tiers.

Best Practices for Full Site Switchover

Oracle recommends the following best practices:

- Maintain the same configuration in primary and standby sites: any changes applied to the primary system must be performed in the secondary system too, so both primary and

secondary systems have the same configuration. For example: a modification in the primary load balancer, any modifications to the operating system, and so on.

- Perform regular switchovers to verify the health of the secondary site.
- Perform any switchover related activity that does not require downtime before you stop the primary servers. For example, the WLS configuration replication based on config_replica.sh script does not require downtime, you can perform it while the primary system is up and running. Other example is to start any shutdown host in the standby site.
- If required to restart the application servers, stop and start the managed servers / mid tiers in parallel.
- The front-end update in DNS is customer dependent. Use a low TTL value in the appropriate DNS entry (at least during the switchover operation) to reduce the time for update. Once the switchover finished, the TTL can be reverted to its original value.
- The OCI load balancer takes some time also to realize that the servers are up and to start sending requests to them. It is usually some seconds, depending on the frequency of the OCI load balancer health checks. Lower the interval used for the checks is, faster it realizes that the servers are up. However, be cautious when you use too low intervals: if the health check is a heavy check, it could overload the back end.

More Information About Full Site Switchover

The previous topics describe full site failover in a generic fashion. For detailed information for full site failover for specific applications refer to the following sources:

- SOA Suite on Oracle Cloud Infrastructure Marketplace Disaster Recovery
- Oracle WebLogic Server for Oracle Cloud Infrastructure Disaster Recovery
- Full Stack Disaster Recovery
- Oracle Cloud Infrastructure Full Stack Disaster Recovery