


Oracle Maximum Availability Architecture (MAA) GoldenGate Hub

A short, solid yellow horizontal line positioned below the title.

Dec 4, 2020
Copyright © 2020, Oracle and/or its affiliates
Confidential: Public Document

DISCLAIMER

This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle software license and service agreement, which has been executed and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

This document is for informational purposes only and is intended solely to assist you in planning for the implementation and upgrade of the product features described. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described in this document remains at the sole discretion of Oracle.

Due to the nature of the product architecture, it may not be possible to safely include all features described in this document without risking significant destabilization of the code.

TABLE OF CONTENTS

Disclaimer	1
Purpose Statement & Intended Audience	3
Introduction	3
Configuration Overview	5
Oracle GoldenGate	5
GoldenGate Hub	6
GoldenGate Hub MAA	6
NGINX Reverse Proxy Server	8
Naming Conventions Used Throughout This Paper	8
Configuration Prerequisites	9
Database Patch Requirements	9
Database Configuration for GoldenGate	9
GoldenGate Database Administrator Account	10
Create Database Role Based Service	11
Oracle Net Services Connectivity	11
GoldenGate Hub Configuration	12
Step 1 - Install Oracle Grid Infrastructure 19c Software	13
Step 2 - Install Oracle Client Software	13
Step 3 - Install GoldenGate Software	14
Step 4 - Install Oracle Standalone Agent Software	14
Step 5 - Install NGINX Reverse Proxy Server	15
Step 6 - Create Application Virtual IP Address	16
Step 7 - Configure ACFS File System Replication	17
Step 8 - Configure Initial GoldenGate Microservices Deployment	25
Step 9 - Create Additional GoldenGate Deployments	30
Step 10 - Register GoldenGate Deployments with Grid Infrastructure Standalone Agent	31
Step 11 - Configure NGINX Reverse Proxy	33
GoldenGate Configuration	36
Creating the Oracle Net Services Client Configuration Files	36
GoldenGate Extract Process Recommendations	37
Oracle Data Guard Data Loss Failover Considerations	37
GoldenGate Replicat Process Recommendations	39
GoldenGate Distribution Path Recommendations	40
Create Autostart Profiles	41
Monitoring GoldenGate Processes	41
Managing Planned and Unplanned Outages for The Goldengate Hub	43
Planned Outages	43
Unplanned Outages of the GoldenGate Hub	45
Conclusion	47
Appendix A: Example SSH Daemon Restart Action Script	48
Appendix B: acfs_primary CRS Action Script	50
Appendix C: acfs_standby CRS Action Script	57
Appendix D: Distribution Path Target Change Script	62
Appendix E: Example GoldenGate Deployment Creation Response Files	64
Appendix F: Troubleshooting	67
Troubleshooting ACFS Replication	67
Troubleshooting GoldenGate	72

PURPOSE STATEMENT & INTENDED AUDIENCE

The intended audience for this document are individuals using Oracle GoldenGate or planning to use Oracle GoldenGate. The document's organization and content assumes familiarity with Oracle Database concepts but is suitable for both individuals well versed in Oracle GoldenGate as well as those that are new to Oracle GoldenGate and replication concepts.

The primary intent of this document is to provide education and guidance in regards to how best to configure a GoldenGate hub to be highly available and tolerant to disasters. Configuring Oracle GoldenGate as a hub is an architectural optimization to offload significant GoldenGate processing and management from the database servers. Ensuring that the hub uses Oracle MAA high availability and disaster recovery best practices is a key prerequisite to an MAA Platinum reference architecture. For more information about MAA platinum solution, see the Oracle Exadata MAA - Platinum Tier Focused Presentation at <https://www.oracle.com/a/tech/docs/exadata-maa-platinum-focused.pdf>.

This paper does not apply to Oracle Cloud Infrastructure (OCI) Marketplace environment.


INTRODUCTION

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. 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 fail over between Oracle RAC nodes and follow Oracle Active Data Guard switchover and failovers.

Using Oracle GoldenGate in a hub configuration moves the GoldenGate software and processes off of the database server, and places them on a separate server located close on the network to the target database. Connectivity between the GoldenGate processes and the databases they operate against is managed with Oracle Net Services.

The Oracle GoldenGate hub configuration provides the following advantages:

- Cross-database versions support
- Cross-platform or cross-endianness support. The source database can be a different platform from the GoldenGate hub (e.g. IBM AIX source database host and Linux GoldenGate hub)
- Offload Oracle GoldenGate software installation, configuration and life cycle management from source and target database systems
- Reduced Oracle GoldenGate resource impact on the source and target database systems
- Support for multiple source and target databases replicating data concurrently
- More than one hub may be used if necessary or for performance reasons (i.e. bi-directional replication between databases in different continents)



Configuring the hub with Maximum Availability Architecture principles in mind, it is possible to create a standby hub that will become the primary hub in the event of a site failure.

CONFIGURATION OVERVIEW

This section introduces the main components of the Oracle GoldenGate hub in an MAA configuration using the GoldenGate Microservices Architecture.

Oracle GoldenGate

GoldenGate provides real-time, log-based change data capture and delivery between homogenous and heterogeneous systems. This technology enables you to construct a cost-effective and low-impact real-time data integration and continuous availability solution.

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.

GoldenGate Microservices Architecture was introduced in GoldenGate release 12.3, as a new administration architecture that provides REST-enabled services as part of the GoldenGate environment. The REST-enabled services provide remote configuration, administration, and monitoring through HTML5 web pages, command-line interfaces, and APIs. Figure 1 shows the GoldenGate Microservices Architecture.

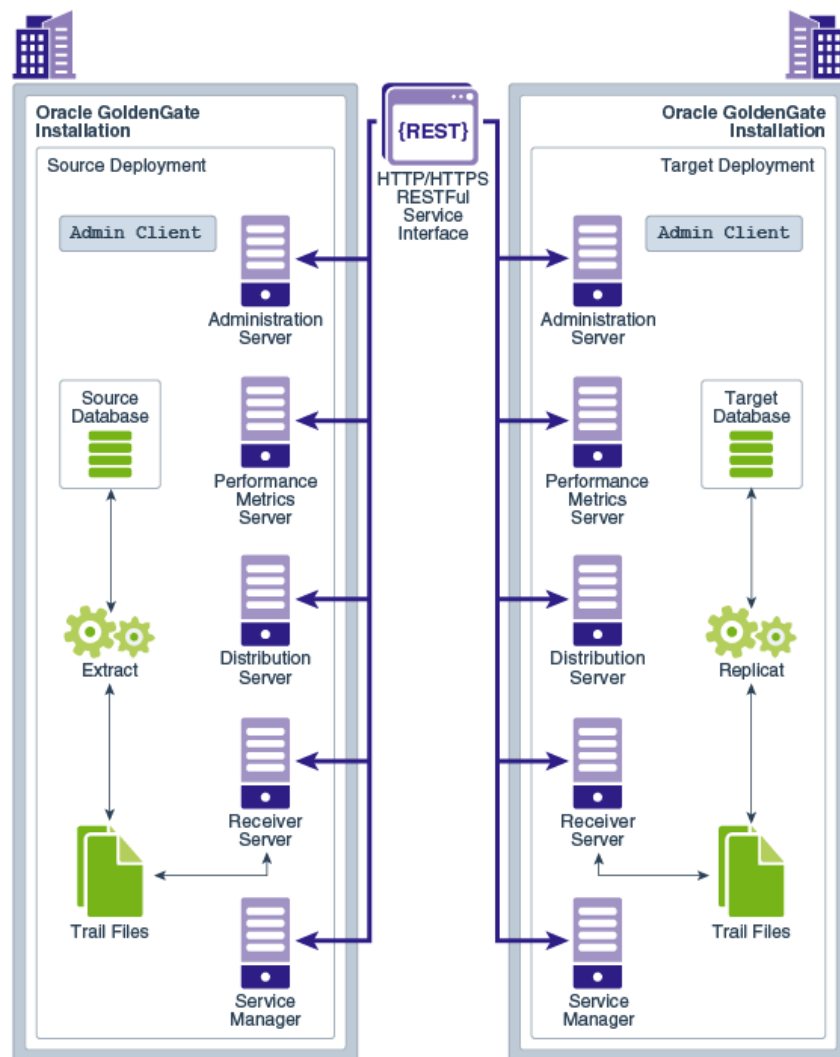


Figure 1: GoldenGate Microservices Architecture

More information about GoldenGate Microservices Architecture can be found in the GoldenGate documentation at:

<https://docs.oracle.com/en/middleware/goldengate/core/19.1/using/getting-started-oracle-goldengate.html#GUID-61088509-F951-4737-AE06-29DAEAD01C0C>

GoldenGate Hub

The GoldenGate Hub is an architectural concept that places the GoldenGate software on a different host than the databases being operated against, as shown in Figure 2. The hub must be in close network proximity to the target database with the expectation that network latency should never exceed 2-3ms, or typically less than 50 kilometers. One advantage of this architecture is that it isolates most of the GoldenGate resource usage from the source and target database servers enabling more system resources for application and database processing. Another advantage is that the GoldenGate configuration is simplified significantly by administering and monitoring the entire GoldenGate infrastructure from a self contained single server, without the need of accessing separate GoldenGate installations on each database server.

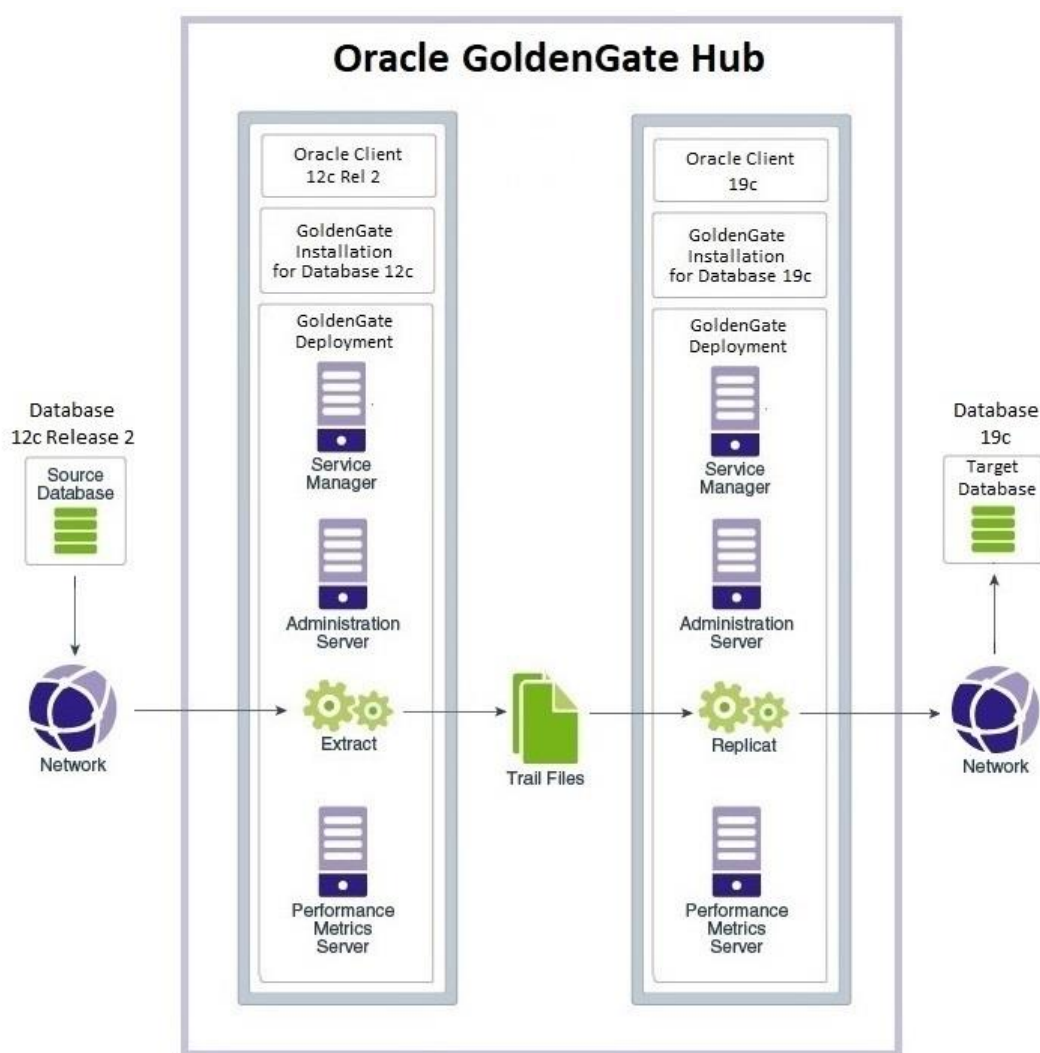


Figure 2: GoldenGate Hub architecture

GoldenGate Hub MAA

In order to provide maximum availability for local and complete disaster failures for the hub, there are additional Oracle software configuration requirements that are shown below in Figure 3.

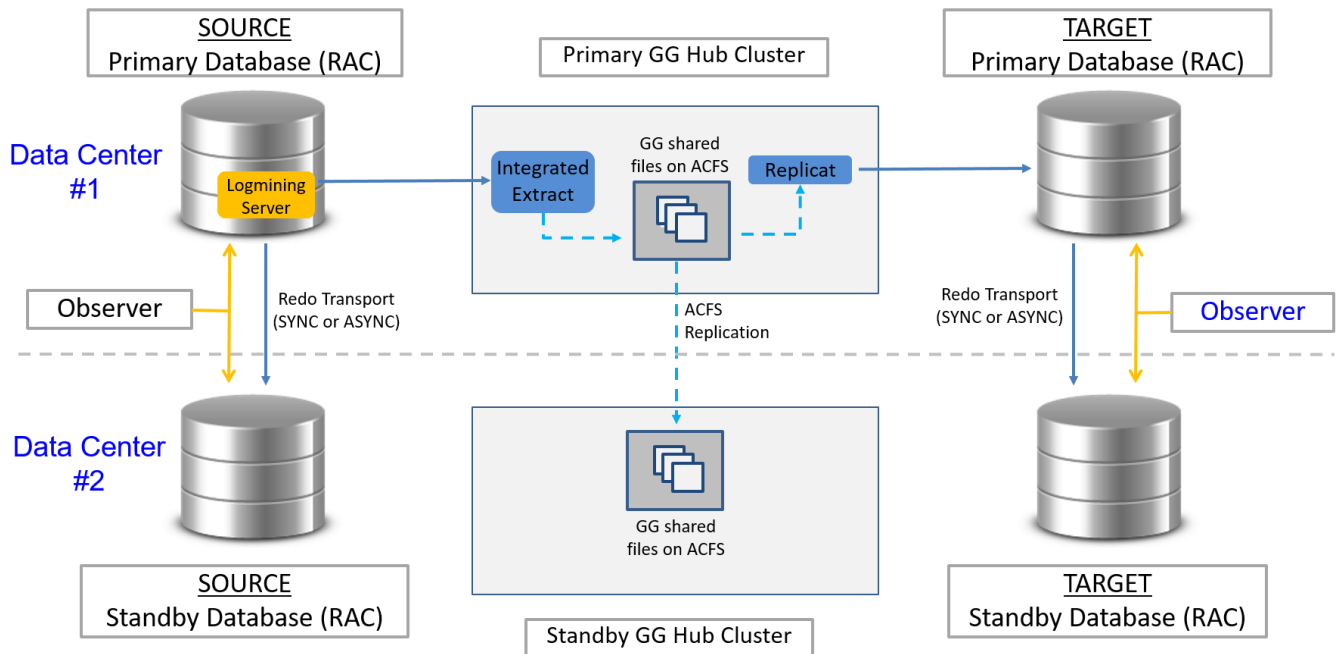


Figure 3: Oracle GoldenGate Hub with MAA

Figure 4 below shows how the GoldenGate hub changes after a data center failure in the environment.

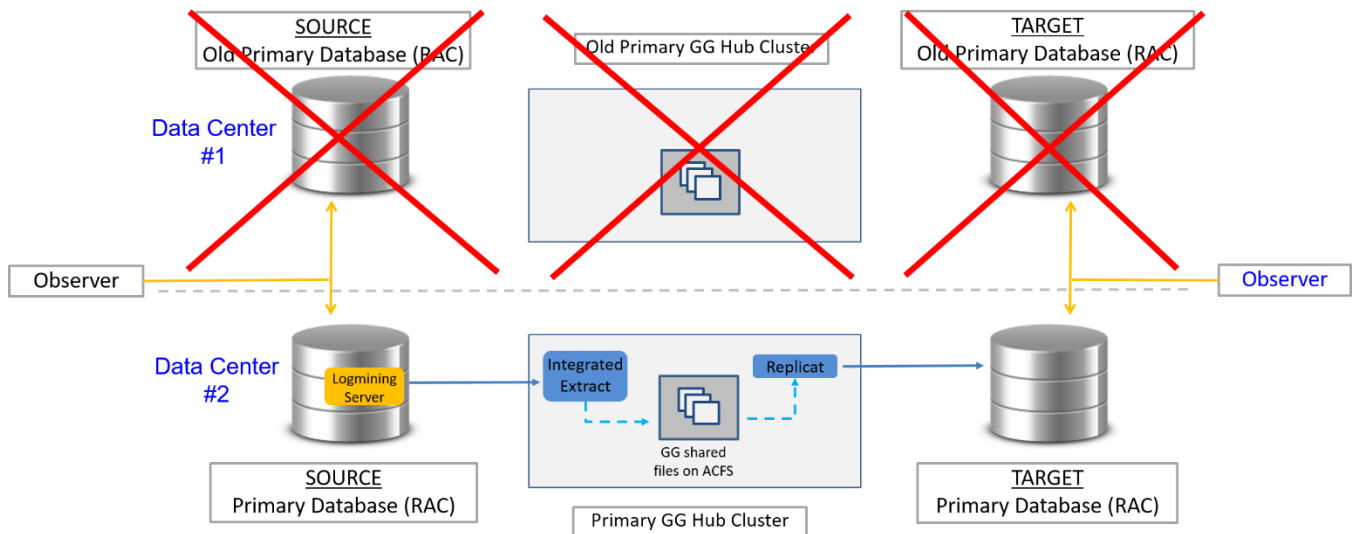


Figure 4: GoldenGate Hub after environment failures

The key hub configuration pre-requisites are:

- Primary and Standby hub servers and target database servers have round trip latency of less 4ms. 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.
- Linux x86-64 Operating System on the hub servers.

- Oracle Grid Infrastructure 19c version 19.7 or higher. When deploying a new GoldenGate hub, it's recommended to use the latest production Grid Infrastructure software.
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 GoldenGate resources can be managed to provide availability in the event of a failure. This minimum release is required to provide the features for seamless integration of ACFS (ASM Cluster File System) Replication and GoldenGate.
- Oracle ACFS to store critical GoldenGate deployment files.
- Oracle ACFS Replication to replicate the critical GoldenGate files to the hub standby server. With ACFS Replication, changes to the GoldenGate shared files, for example checkpoint and trail files, are automatically replicated to a standby file system.
- Oracle Grid Infrastructure Standalone Agent (XAG) version 10.2 or higher.
The Oracle Grid Infrastructure Standalone Agent leverages the Oracle Grid Infrastructure components to provide integration between GoldenGate and its dependent resources, such as the database, network and file system. The agent also integrates GoldenGate with Oracle Data Guard so that GoldenGate is restarted on the new primary database following a role transition.
- GoldenGate 19c version 19.1.0.0.4 Microservices or higher.
GoldenGate Microservices Architecture provides REST-enabled services as part of the GoldenGate environment. The REST-enabled services provide remote configuration, administration, and monitoring through HTML5 web pages, command-line interfaces, and APIs.

NGINX Reverse Proxy Server

The GoldenGate reverse proxy feature allows a single point of contact for all of the GoldenGate microservices associated with an GoldenGate deployment. Without reverse proxy, the 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 might be `http://gghub.example.com:9101`, the administration server of deployment #2 could be `https://gghub.example.com:9111`, and so on.

With reverse proxy, port numbers are not required to connect to the microservices, as they are replaced with the deployment name. Using the previous example, to connect to the Service Manager use the URL `https://gghub.example.com`, the Admin Server of deployment #1 (named Source) would use `https://gghub.example.com/SOURCE/adminsrvr`, and the administration server of deployment #2 (named Target) would be `https://gghub.example.com/TARGET/adminsrvr`.

Reverse proxy is recommended and used by default with GoldenGate from the Oracle Cloud Marketplace to ensure easy access to microservices and provide enhanced security and manageability.

Configuration of the NGINX proxy server is covered later in this paper.

Naming Conventions Used Throughout This Paper

Throughout this paper, examples are provided for REST API endpoints to manage the GoldenGate configuration using two deployments (SOURCE and TARGET). The REST API is used so that the commands can easily be integrated into an automated configuration script. The script can be run locally or remotely from any server with access to the hub with curl and Python installed. Alternatively, you can use the GoldenGate Admin Client commands to manage the hub. Admin Client is a standalone command-line interface used to create and manage GoldenGate replication. Refer to the *Command Line Interface Reference for GoldenGate* for Admin Client commands.

<https://docs.oracle.com/en/middleware/goldengate/core/19.1/qclir/index.html>

Here is an example of key arguments given in the GoldenGate REST APIs used throughout this brief:

```
$ curl -s -K access.cfg https://<GG Hub>/<Deployment  
Name>/adminsrvr/services/v2/credentials/goldengate -XGET | python -m json.tool
```

- access.cfg: To prevent the GoldenGate administrator account name and password from being exposed on the command line, it is recommended that you include the user name and password in a configuration file, which is read by curl, as shown here.

```
user = "oggadmin:password"
```

- GG Hub: The hostname or IP address of the hub server. For example, gghub-server.
- Deployment Name: This is the name of the GoldenGate deployment. For example, SOURCE or TARGET.

Example:

```
$ curl -s -K access.cfg  
https://gghub.example.com/SOURCE/adminsrvr/services/v2/credentials/goldengate -XGET | python -m  
json.tool
```

The example Extract name used in this paper is EXT1 and the Replicat is called REP1.

The REST calls can be made from either the hub, or any machine, that can access the GoldenGate hub through the HTTPS protocol.

CONFIGURATION PREREQUISITES

Database Patch Requirements

The minimum database version that is supported with GoldenGate is Oracle Database 11g Release 2 (11.2.0.4). It is a best practice to apply the latest bundle patch and PSU (Patch Set Update) on both the source and target databases. The full recommended patch list can be found in My Oracle Support note 2193391.1, which covers database versions 11.2.0.4 onwards.

It is recommended that you also apply patch 28849751 to the source database if network round trip latency between the source database and hub is greater than 8ms, and if the latest database bundle patch or PS/CPU (Critical Patch Update) does not include it.

To check if your current database software includes patch 28849751 run the following command.

```
$ $ORACLE_HOME/OPatch/patch lsinventory|grep 28849751  
  
28803345, 28819640, 28849751, 28852691, 28856060, 28891984, 28951382
```

Database Configuration for GoldenGate

The source database prerequisites are described in the MAA technical brief "Oracle GoldenGate Performance Best Practices" at <https://www.oracle.com/technetwork/database/availability/maa-gg-performance-1969630.pdf>. The key prerequisites are listed below.

Source Database Prerequisites

- Enable ARCHIVELOG mode for the database.
- Enable database force logging to ensure that all changes are found in the redo by the GoldenGate Extract process.
- Enable database minimal supplemental logging. Additional schema level supplemental logging for replicated objects also required.
- Configure the streams pool with the initialization parameter STREAMS_POOL_SIZE.
- Enable GoldenGate replication by enabling the initialization parameter ENABLE_GOLDENGATE_REPLICATION.
- Install the UTL_SPADV/UTL_RPADV package for integrated Extract performance analysis.

Target Database Prerequisites

- Enable GoldenGate replication by enabling the initialization parameter ENABLE_GOLDENGATE_REPLICATION.

- Configure the streams pool with the initialization parameter `STREAMS_POOL_SIZE`, if GoldenGate integrated parallel Replicat is used (refer to the section [GoldenGate Replicat Process Recommendations](#)).
- Install the `UTL_SPADV/UTL_RPADV` package for integrated Extract performance analysis.

GoldenGate Database Administrator Account

If the source database is currently part of a GoldenGate configuration the GoldenGate administrator account may already exist. If the user already exists, you need to confirm the permissions are correctly granted, and if not grant them.

If no GoldenGate database administrator user account exists, the user must be created. The recommended source database username is `GGADMIN` for single tenant and PDBs where not all PDBs are being replicated, and `C##GGADMIN` for a CDB database and PDBs where a single PDB exists or all PDBs are being replicated. The target database recommended username is `GGADMIN`.

If a GoldenGate administrator user already exists, it should be used for the database GoldenGate configuration. Make sure the permissions listed below are granted to the current GoldenGate administrator account.

Throughout this technical brief the database GoldenGate administrator account is named `GGADMIN`.

Use the following example to create a new GoldenGate administrator account for a single tenant source database, a subset of PDBs that exist in the multitenant database, and for the target database.

```
SQL> create user ggadmin identified by <password>
      default tablespace users temporary tablespace temp;
SQL> grant connect, resource to ggadmin;
SQL> grant select any dictionary to ggadmin;
SQL> grant create view to ggadmin;
SQL> grant execute on dbms_lock to ggadmin;
SQL> exec dbms_goldengate_auth.GRANT_ADMIN_PRIVILEGE('ggadmin');
```

If you are replicating all PDBs within a multitenant source database, use the following example command to create the common user in the CDB and all PDBs:

```
SQL> create user c##ggadmin identified by <password>
      default tablespace users temporary tablespace temp CONTAINER=ALL;
SQL> grant connect, resource to c##ggadmin CONTAINER=ALL;
SQL> grant select any dictionary to ggadmin CONTAINER=ALL;
SQL> grant create view to c##ggadmin CONTAINER=ALL;
SQL> grant execute on dbms_lock to c##ggadmin CONTAINER=ALL;
SQL> exec dbms_goldengate_auth.GRANT_ADMIN_PRIVILEGE('c##ggadmin',container=>'all');
```

NOTE: The default and temporary tablespace names **MUST** exist in the CDB and all PDBs.

If you are replicating a subset of the PDBs within a multitenant source database, use the following example command to create the user in the CDB:

```
SQL> create user c##ggadmin identified by <password>
      default tablespace users temporary tablespace temp;
SQL> grant connect, resource to c##ggadmin;
SQL> grant select any dictionary to ggadmin;
SQL> grant create view to c##ggadmin;
SQL> grant execute on dbms_lock to c##ggadmin;
SQL> exec dbms_goldengate_auth.GRANT_ADMIN_PRIVILEGE('c##ggadmin',container=>'<PDB name>');
```

Create Database Role Based Service

If the GoldenGate source and target databases are running the recommended configuration on an Oracle RAC cluster with Oracle Data Guard, a role-based service needs to be created that will allow the GoldenGate Extract or Replicat processes to connect to the correct Data Guard primary database instance. Run the following command on both the primary and standby Data Guard clusters to create the role based service:

```
$ srvctl add service -db <database_service_name> -service oggsource -role PRIMARY  
-preferred <instance_name_1>,<instance_name_2>
```

Example:

```
$ srvctl add service -db GGS2 -service oggsource -role PRIMARY -preferred GGS21,GGS22
```

NOTE: A database service is still required if not using Oracle Data Guard, simply omit the `-role PRIMARY` parameter.

The service should be started before GoldenGate is configured in a later step, using the following commands:

```
$ srvctl status service -d GGS2 -s oggsource  
  
$ srvctl start service -d GGS2 -i GGS21 -s oggsource
```

NOTE: The role based service must be created on both the source and target database Data Guard primary and standby RAC clusters.

Oracle Net Services Connectivity

Oracle Net connectivity should be optimized to provide the best performance for the remote GoldenGate Extract running on the hub.

It is recommended that you set the source database listener to use a higher SDU size to improve performance with a remote GoldenGate Extract. Please note that even though the listener is set at the maximum possible SDU size, the smallest size is used if requested by the client. For example, if the listener sets SDU to 2MB but the client requests the default 8KB SDU, the connection to the database uses an 8KB SDU size. The maximum value for the SDU size for Oracle Database 11g Release 2 is 64KB (65536 bytes), and for later database releases the maximum value is 2MB (2097152 bytes).

It is recommended that you use Oracle Net Services or Secure Sockets Layer (SSL) encryption for connections to the source database. Refer to the *Oracle Database Security Guide* for more information about SSL with Oracle Net.

<https://docs.oracle.com/en/database/oracle/oracle-database/19/dbseg/configuring-secure-sockets-layer-authentication.html#GUID-6AD89576-526F-4D6B-A539-ADF4B840819F>

The following example `sqlnet.ora` and `listener.ora` files are configured with SSL and an increased SDU size for Oracle Database 12g Release 1 and later:

sqlnet.ora

```
SQLNET.IGNORE_ANO_ENCRYPTION_FOR_TCPS = TRUE  
SQLNET.WALLET_OVERRIDE = FALSE  
SQLNET.EXPIRE_TIME = 10  
WALLET_LOCATION = (SOURCE=(METHOD=FILE) (METHOD_DATA=(DIRECTORY=/u01/oracle/tcps_wallets)))  
SSL_VERSION = 1.2  
  
# Parameters required for Net encryption if not using SSL Authentication, replacing  
# the above parameters:  
# SQLNET.ENCRYPTION_SERVER = accepted  
# SQLNET.ENCRYPTION_TYPES_SERVER= (AES256)  
  
DEFAULT_SDU_SIZE = 2097152
```

listener.ora

```
GGSOURCE =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (SDU = 2097152)
      (ADDRESS = (PROTOCOL = TCPS) (HOST = <source database host>) (PORT = 2484))
    )
  )
SID_LIST_GGSOURCE =
  (SID_LIST =
    (SID_DESC =
      (SDU = 2097152)
      (SID_NAME = <ORACLE SID>)
      (ORACLE_HOME = <ORACLE_HOME>)
    )
  )
)
```

If the source database is on Oracle RAC, configure the listener on all of the cluster nodes running instances for the database. This way, if one instance goes down, the service can migrate to a surviving instance. Configure the SSL wallet on all Oracle RAC nodes.

To start, stop, and get the status of the listener, make sure the following environment variables are set and then issue the listed commands:

```
export ORACLE_HOME=<oracle home directory>
export PATH=$PATH:$ORACLE_HOME/bin
export TNS_ADMIN=<TNS admin directory>
```

To start the listener:

```
$ lsnrctl start GGSOURCE
```

To stop the listener:

```
$ lsnrctl stop GGSOURCE
```

To get the current status of the listener:

```
$ lsnrctl status GGSOURCE
```

GOLDENGATE HUB CONFIGURATION

The hub requires four software installations and configuration:

- Oracle Grid Infrastructure 19c, version 19.7 or higher
- Oracle Client Software to match source and target Oracle Database versions (minimum version is 11.2.0.4)
- GoldenGate software to match source and target Oracle Database versions
- NGINX reverse proxy server used by the GoldenGate Microservices deployment connectivity

The following sections provide details about the required software installations.

NOTE: All instructions must be carried out on both the primary and standby hub servers.

Step 1 - Install Oracle Grid Infrastructure 19c Software

After installing Oracle Grid Infrastructure 19c on both the primary and standby hub servers, download and apply the latest release update. The minimum required release update version is 19.7.

You MUST apply patch 31697904 to the Oracle Grid Infrastructure software home to ensure that you have the latest ACFS replication functionality to handle unplanned file system outages, which is required for the solution presented in this technical brief. This patch can be downloaded from My Oracle Support.

Step 2 - Install Oracle Client Software

GoldenGate requires libraries that are installed as part of the Oracle Client Runtime installation. The client software version installed must be the same release number as the database that GoldenGate will connect to. For example, if the source is Oracle Database 12c Release 2, and the target is Oracle Database 19c, two different client software installations are required: one for Oracle Database 12c and another for Oracle Database 19c. If the source and target database releases are the same, a single Oracle client software installation is required.

If the source database is not running on the same operating system as the hub, install the GoldenGate software for the hub platform operating system, which is Linux.

*NOTE: The Oracle Client Software **MUST** be installed in matching ORACLE_HOME directories on both the primary and standby hub servers. Do NOT use the Oracle instant client software because it does not contain all of the required libraries.*

Download the Oracle client software from <https://edelivery.oracle.com>. On Oracle Software Delivery Cloud select the Release category and search for "Oracle Database Client".

During installation, select the Runtime installation type, as shown in the example below.

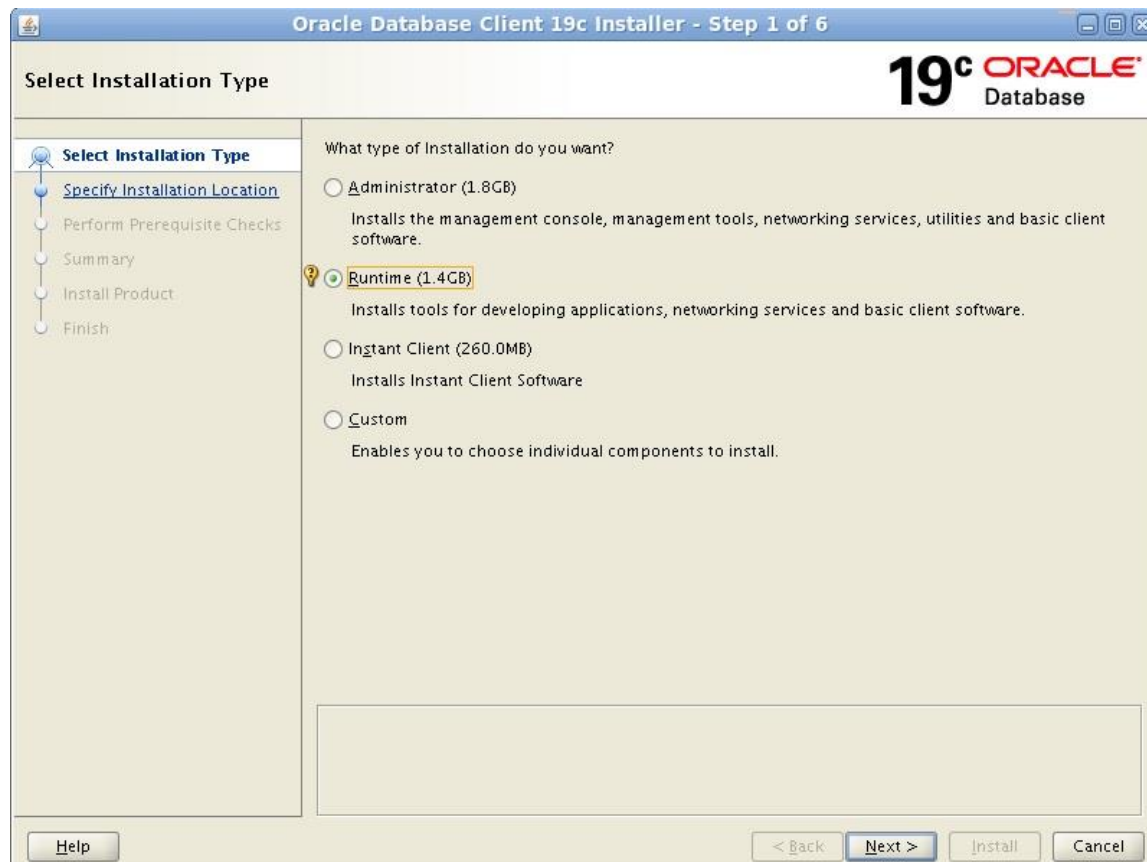


Figure 4: Oracle Client Software Installation

Install the Oracle client software that matches both the source and target database releases into separate ORACLE_HOME directories.

Step 3 - Install GoldenGate Software

You must install the GoldenGate software release that supports the source and target Oracle Database releases. The GoldenGate software compatibility matrix can be found at <https://www.oracle.com/technetwork/middleware/ias/downloads/fusion-certification-100350.html>

Download the latest GoldenGate software from <https://www.oracle.com/middleware/technologies/goldengate-downloads.html>. It is recommended that you install the latest available release of GoldenGate, which is currently 19c Release 1 (19.1.0.0.4). Using GoldenGate 19c Release 1 allows the cross-endian extract where the operating system running GoldenGate Extract is a different endianness from the source database platform.

If the source and target database releases are different, make sure you install the GoldenGate software twice, once for each database release. Below, Figure 5 shows the software installer option for the database releases.

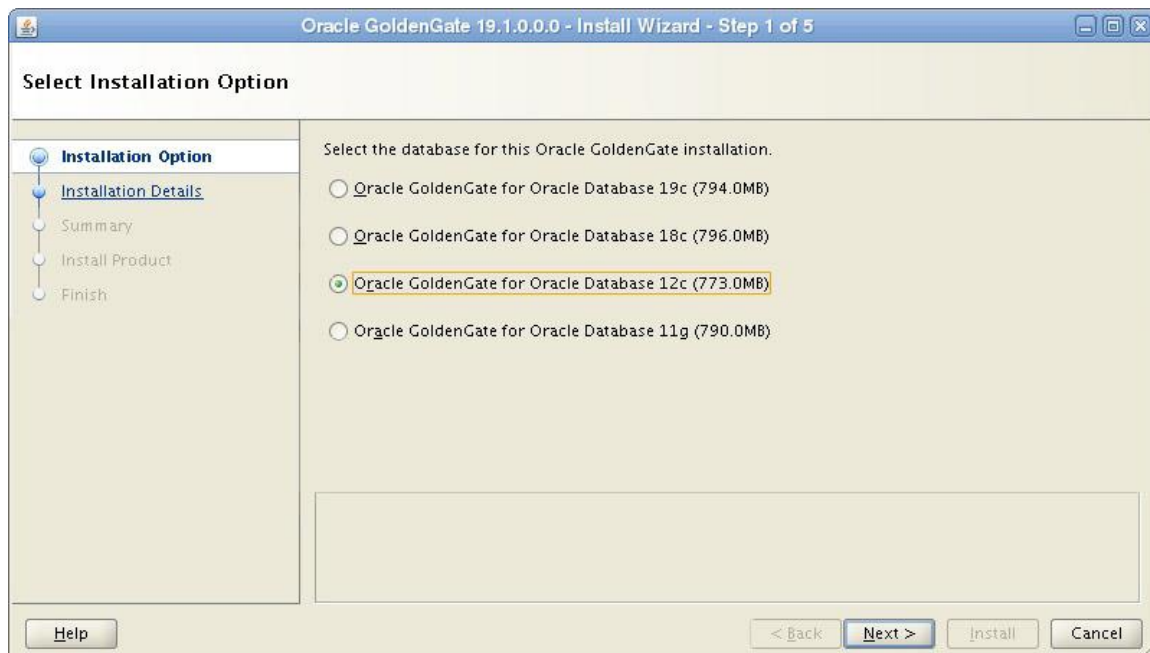


Figure 5. Select the GoldenGate release for your Oracle Database version

It is important that you install the GoldenGate software in separate directories, based on the database version they support.

NOTE: Make sure to place each GoldenGate software installation in the same directory path on each primary and standby Oracle RAC nodes.

Step 4 - Install Oracle Standalone Agent Software

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 RAC nodes and Data Guard primary databases.

- Install the Oracle Grid Infrastructure Standalone Agent

Download the latest XAG software from <http://www.oracle.com/technetwork/database/database-technologies/clusterware/downloads/xag-agents-downloads-3636484.html>

When using GoldenGate Microservices Architecture, it is a requirement to use XAG version 10.2 or newer.

The XAG software **must** be installed in a directory that is outside of the Oracle Grid Infrastructure home directory to avoid conflicts with software patching. XAG needs to be installed in the same directory path on all RAC nodes in the cluster where GoldenGate is installed.

For example, as the `oracle` user:

```
$ ./xagsetup.sh --install --directory /u01/oracle/xag --all_nodes
```

- Add the location of the installed XAG software to the PATH variable so it gets set automatically for the oracle user.

```
$ cat .bashrc
export PATH=/u01/oracle/xag/bin:$PATH
```

NOTE: It is important to make sure the XAG directory is specified BEFORE the Grid Infrastructure directory for the PATH variable to ensure the correct `agctl` binary is found.

Step 5 - Install NGINX Reverse Proxy Server

The GoldenGate reverse proxy feature allows a single point of contact for all of the GoldenGate microservices associated with a 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. 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 administration server of deployment #2 may be `https://gghub.example.com:9111`, and so on.

With reverse proxy, port numbers are not required to connect to the microservices, because they are replaced with the deployment name. Using the previous example, to connect to the Service Manager use the URL `https://gghub.example.com`, the Admin Server of deployment #1 (named Source) would use `https://gghub.example.com/SOURCE/adminsrvr`, and the administration server of deployment #2 (named Target) would be `https://gghub.example.com/Target/adminsrvr`.

Reverse proxy is recommended to ensure easy access to microservices and provide enhanced security and manageability.

The GoldenGate reverse proxy feature uses NGINX reverse proxy. The following instructions describe how to configure the reverse proxy.

1. Check that NGINX is not already installed (as root).

```
$ sudo rpm -qa |grep nginx
```

If not installed, nothing is returned.

If NGINX is installed, the output is similar to the following.

```
nginx-mod-http-xslt-filter-1.12.2-2.el7.x86_64
nginx-mod-http-image-filter-1.12.2-2.el7.x86_64
nginx-filesystem-1.12.2-2.el7.noarch
nginx-mod-mail-1.12.2-2.el7.x86_64
nginx-mod-http-perl-1.12.2-2.el7.x86_64
nginx-1.12.2-2.el7.x86_64
nginx-all-modules-1.12.2-2.el7.noarch
nginx-mod-http-geoip-1.12.2-2.el7.x86_64
nginx-mod-stream-1.12.2-2.el7.x86_64
```


2. If NGINX is not already installed, install it. The following instructions assume the YUM (Yellowdog Updated Modified) is installed and configured for easier package installation.

```
$ sudo yum install epel-release
$ sudo yum update
$ sudo yum install nginx
```

3. Start NGINX.

```
$ sudo systemctl start nginx
```

4. Verify that NGINX is running.

```
$ curl -I 127.0.0.1

HTTP/1.1 200 OK
Server: nginx/1.12.2
Date: Wed, 4 Apr 2019 21:48:43 GMT
Content-Type: text/html
Content-Length: 3700
Last-Modified: Wed, 4 Apr 2019 05:06:50 GMT
Connection: keep-alive
ETag: "5abdc5ea-e74"
Accept-Ranges: bytes
```

5. Enable NGINX to auto-start when the machine starts

To enable auto-starting of NGINX, run the following command as the root user:

```
# systemctl enable nginx
```

Complete documentation for NGINX installation can be found at <https://docs.nginx.com/nginx/admin-guide/installing-nginx/installing-nginx-open-source/>.

Step 6 - Create Application Virtual IP Address

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.

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 NAME,USR_ORA_SUBNET -w "TYPE = ora.network.type" |sort | uniq
NAME=ora.net1.network
USR_ORA_SUBNET=10.231.41.0
```

The net1 in NAME=ora.net1.network indicates this is network 1.

- 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.

Create the VIP CRS resource (as `root`), using the following example command.

```
# appvipcfg create -network=1 -ip=12.123.12.123 -vipname=gg_vip_prmy -user=oracle
```

Refer to the *Oracle Clusterware Administration and Deployment Guide* for further information on creating an Application VIP.

<https://docs.oracle.com/en/database/oracle/oracle-database/19/cwadd/making-applications-highly-available-using-oracle-clusterware.html#GUID-AFA2B0A1-479F-4DAB-A9D0-226C30260DD8>

After the VIP is created, configure the ssh daemon to listen for incoming connections on the VIP address on the ACFS primary and standby clusters by adding the VIP address to the `/etc/ssh/sshd_config` file and restarting the ssh daemon, as shown here.

Add the following to the `/etc/ssh/sshd_config` file:

```
ListenAddress <VIP address>
```

Restart the ssh daemon:

```
# /bin/systemctl restart sshd
```

Step 7 - Configure ACFS File System Replication

Configuring ACFS file system replication consists of the following substeps:

- Create the file system on the primary and standby servers
- Create the CRS dependency between ACFS and an application VIP
- Create SSH daemon CRS resource
- Enable ACFS replication
- Create ACFS replication CRS action scripts
- Test ACFS switchovers and failovers

A. Create the File System on the Primary and Standby Servers

A file system of the same size must be created on both the primary and standby clusters. The file systems should be sized to permit storage of up to 12 hours of trail files to 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.

1. Create the same mount point on all the primary and standby RAC nodes where ACFS can be mounted.

Example (as `root`):

```
# mkdir /mnt/acfs_rep1
# chown -R oracle:dba /mnt/acfs_rep1
```

2. Using ASMCMD, create the file system volume (as `oracle`):

```
ASMCMD [+] > volcreate -G datac1 -s 1200G ACFS_REP1
ASMCMD [+] > volinfo --all

Diskgroup Name: DATA1
  Volume Name: ACFS_REP1
  Volume Device: /dev/asm/acfs_rep1-89
  State: ENABLED
  Size (MB): 1228800
  Resize Unit (MB): 64
```

```
Redundancy: MIRROR
Stripe Columns: 8
Stripe Width (K): 1024
```

3. Create the file system (as `oracle`):

```
$ /sbin/mkfs -t acfs /dev/asm/acfs_repl-89
```

4. Register the file system with Oracle Grid Infrastructure (as `root`):

```
# srvctl add filesystem -device /dev/asm/acfs_repl-89 -volume ACFS_REP1 -diskgroup DATA1 -path
/mnt/acfs_repl -user oracle -node <node1>,<node2> -autostart NEVER
-mountowner oracle -mountgroup oinstall -mountperm 755
```

5. Test mounting and relocating the ACFS mount between RAC nodes (as `oracle`):

```
$ srvctl start filesystem -d /dev/asm/acfs_repl-89
$ srvctl status filesystem -d /dev/asm/acfs_repl-89
$ srvctl relocate filesystem -diskgroup DATA1 -volume acfs_repl
$ srvctl status filesystem -d /dev/asm/acfs_repl-89
```

B. Create CRS Dependencies Between ACFS and an Application VIP

To ensure that the file system is mounted on the same Oracle RAC node as the VIP, add the VIP CRS resource as a dependency to the ACFS resource, using the following example commands.

NOTE: Make sure the following commands are run on both the primary and standby hub clusters.

1. Stop the VIP resource (as `oracle`).

```
$ crsctl status resource gg_vip_prmy
```

2. Determine the current start and stop dependencies of the VIP resource (as `oracle`).

```
$ crsctl stat res gg_vip_prmy -f|grep _DEPENDENCIES

START_DEPENDENCIES=hard(ora.net1.network) pullup(ora.net1.network)
STOP_DEPENDENCIES=hard(intermediate:ora.net1.network)
```

3. Modify the start and stop dependencies of the VIP resource (as `root`).

```
# crsctl modify res gg_vip_prmy -attr
"START_DEPENDENCIES='hard(ora.net1.network,ora.datacl.acfs_repl.acfs) pullup(ora.net1.network)
pullup:always(ora.datacl.acfs_repl.acfs)'"

# crsctl modify res gg_vip_prmy -attr
"STOP_DEPENDENCIES='hard(intermediate:ora.net1.network,ora.datacl.acfs_repl.acfs)'"
```

NOTE: Substitute `acfs_repl` for the correct ACFS volume name.

4. Restart the VIP (as `oracle`).

```
$ crsctl start resource gg_vip_prmy

CRS-2672: Attempting to start 'ora.datacl.acfs_repl.acfs' on 'node1'
CRS-2676: Start of 'ora.datacl.acfs_repl.acfs' on 'node1' succeeded
CRS-2672: Attempting to start 'gg_vip_prmy' on 'node1'
CRS-2676: Start of 'gg_vip_prmy' on 'node1' succeeded
```

```
$ crsctl status resource gg_vip_prmy

NAME=gg_vip_prmy
TYPE=app.appvtypex2.type
TARGET=ONLINE
STATE=ONLINE on node1
```

5. Test relocation of the VIP and ACFS mount.

```
$ crsctl relocate resource gg_vip_prmy -f

Or

$ srvctl relocate filesystem -diskgroup DATA1 -volume acfs_rep1 -force

$ crsctl status resource gg_vip_prmy

NAME=gg_vip_prmy
TYPE=app.appvtypex2.type
TARGET=ONLINE
STATE=ONLINE on node2

$ srvctl status filesystem -diskgroup DATA1 -volume acfs_rep1

ACFS file system /mnt/acfs_rep1 is mounted on nodes node2
```

C. Create 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.

1. Use the example CRS action script in Appendix B to restart the ssh daemon. Place the script in the same location on all primary and standby ACFS file system nodes.
2. As the `root` user, create the CRS resource using the following example command.

```
crsctl add resource sshd_restart \
-type cluster_resource \
-attr "ACL='owner:root:rw,group:oinstall:r-x,user:oracle:rw',
ACTION_SCRIPT=/u01/oracle/scripts/sshd_restart.scr,CHECK_INTERVAL=0,AUTO_START='always',
START_DEPENDENCIES='hard(gg_vip_prmy) pullup:always(gg_vip_prmy)
attraction(gg_vip_prmy)',STOP_DEPENDENCIES='hard(gg_vip_prmy)'"
```

3. Replace the `gg_vip_prmy` VIP name with the VIP name created in [Step 6 - Create Application Virtual IP Address](#).
4. Test the CRS resource on the current primary and standby ACFS nodes using the following commands.

```
$ crsctl stat res sshd_restart

NAME=sshd_restart
TYPE=cluster_resource
TARGET=OFFLINE
STATE=OFFLINE

$ crsctl start res sshd_restart

CRS-2672: Attempting to start 'sshd_restart' on 'node1'
CRS-2676: Start of 'sshd_restart' on 'node1' succeeded

$ ls -lrt /tmp/sshd_restarted
```

```
-rw-r--r-- 1 root root 0 Jun 29 12:33 /tmp/sshd_restarted

$ crsctl stop res sshd_restart
CRS-2673: Attempting to stop 'sshd_restart' on 'node1'
CRS-2677: Stop of 'sshd_restart' on 'node1' succeeded

$ ls -lrt /tmp/sshd_restarted
ls: cannot access /tmp/sshd_restarted: No such file or directory
```

D. Enable ACFS Replication

Complete the following three steps to enable ACFS Replication.

1. Configure SSH Connectivity Between the Primary and Standby Hosts

ACFS snapshot-based replication uses `ssh` to transfer the snapshots from between the primary and standby hosts. The replication process uses the `root` user on the primary node to connect to a minimally-privileged user on the standby node.

To configure the `ssh` connectivity between the primary and standby nodes follow the instructions provided in the Storage Administrators Guide at

<https://docs.oracle.com/en/database/oracle/oracle-database/20/acfsg/configure-acfs-replication.html#GUID-C1CB3914-E278-473A-ACCC-FD1BA7CDC0B4>

Be sure to complete the configuration between all primary and standby Oracle RAC nodes. Connectivity in the opposite direction should also be configured ready for ACFS replication role reversal.

Once `ssh` configuration is completed, test connectivity between all primary to standby nodes, and in the reverse direction using both `ssh` as the `root` user, and with the `acfsutil` command.

Example (as `root`):

```
# ssh oracle@snode1 uname -a
Linux snode1.example.com 4.1.12-124.26.12.el7uek.x86_64 #2 SMP Wed May 8 22:25:03 PDT 2019 x86_64
x86_64 x86_64 GNU/Linux

# ssh oracle@snode1.example.com uname -a
# ssh oracle@snode2 uname -a
# ssh oracle@snode2.example.com uname -a
# ssh oracle@clust2-vip1 uname -n
# ssh oracle@clust2-vip1.example.com uname -n

## Repeat the above ssh tests as oracle user that must also succeed.

# acfsutil repl info -c -u oracle snode1 snode2 clust2-vip1 /mnt/acfs_repl
A valid 'ssh' connection was detected for standby node snode1 as user oracle.
A valid 'ssh' connection was detected for standby node snode2 as user oracle.
A valid 'ssh' connection was detected for standby node clust2-vip1 as user oracle.
```

NOTE: Make sure the connectivity is verified between all primary file system 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.

2. Initialize Replication on the Standby and Primary File Systems

On the standby node where ACFS is currently mounted (as `oracle`):

```
$ /sbin/acfsutil repl init standby -u oracle /mnt/acfs_repl
```

On primary node where ACFS is currently mounted (as oracle):

```
$ /sbin/acfsutil repl init primary -C -p oracle@clust1-vip1 -s oracle@clust2-vip1 -m /mnt/acfs_repl /mnt/acfs_repl
```

During the standby file system initialization, the primary file system contents are copied to the standby node. During this time, you can monitor the initialization progress using the following command:

```
$ /sbin/acfsutil repl info -c -v /mnt/acfs_repl | grep Status  
Status: Sending initial copy
```

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.

Refer to the Oracle Storage Administrators Guide for more information about using acfsutil to initialize replication on the standby and primary file systems.

<https://docs.oracle.com/en/database/oracle/oracle-database/19/ostmg/acfs-commands-replication.html#GUID-E96CD9A2-00BD-4673-8FCC-DBB28A9E2DCD>

3. Verify and Monitor the Replicated File System

The status of ACFS Replication on both the primary and standby file systems can be obtained by running the following commands.

On the primary host:

```
$ acfsutil repl util verifystandby /mnt/acfs_repl  
verifystandby return code: 0
```

On the standby host:

```
$ acfsutil repl util verifyprimary /mnt/acfs_repl  
verifyprimary return code: 0
```

Both commands will return a value of 0 (zero) if there are no problems detected. If a non-zero value is returned, refer to [Appendix A – Troubleshooting Oracle ACFS Replication](#) for monitoring, diagnosing, and resolving common issues with ACFS Replication before continuing.

To monitor the current status of ACFS Replication, use the following example command on both the primary and standby hubs.

```
$ /sbin/acfsutil repl info -c -v /mnt/acfs_repl  
  
Site: Primary  
Primary hostname: clust17-vip1  
Primary path: /mnt/acfs_repl  
Primary status: Running  
Background Resources: Active  
  
Standby connect string: oracle@clust2-vip1  
Standby path: /mnt/acfs_repl  
Replication interval: 0 days, 0 hours, 0 minutes, 0 seconds  
Sending primary as of: Tue Aug 04 12:24:03 2020  
Status: Sending incremental differences  
Retries made: 0  
Last send started at: Tue Aug 04 12:24:03 2020  
Last send completed at: In progress  
Next send starts at: Tue Aug 04 12:24:03 2020  
Replicated tags:
```

```
Data transfer compression:      Off
ssh strict host key checking:    On
Debug log level:                3
Replication ID:                 0xadc39f3b
```

E. Create 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_oracle.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 cluster nodes.

On both the primary and standby file system clusters there are two scripts required, one to monitor the local primary file system and verify the remote standby file systems is available, and one to monitor the local standby file system and check remote primary file systems' availability. Example scripts are provided in [Appendix B](#) and [Appendix C](#) to implement the ACFS monitoring, but you must edit them to suit your environment.

Below are descriptions of the two example CRS action scripts.

acfs_primary

The `acfs_primary` 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 GoldenGate can start processes on the primary GoldenGate hub. If the standby file system is not accessible by the primary, the example script in [Appendix B](#) makes multiple attempts to verify the standby file system, printing the following warning messages in the `crsd_scriptagent_oracle.trc` trace file:

```
2020-11-16 15:15:01.506:[acfs_primary]{1:40360:26097} [check] WARNING: STANDBY not accessible
(attempt 1 of 3))
2020-11-16 15:15:14.535:[acfs_primary]{1:40360:26097} [check] WARNING: STANDBY not accessible
(attempt 2 of 3))
2020-11-16 15:15:27.611:[acfs_primary]{1:40360:26097} [check] WARNING: STANDBY not accessible
(attempt 3 of 3))
2020-11-16 15:15:27.611:[acfs_primary]{1:40360:26097} [check] WARNING: Problem with STANDBY file
system (error: 222)
```

When the primary and standby file systems are online and available, the following messages are output into the trace file:

```
2020-11-04 21:35:48.588:[acfs_primary]{1:20735:25070} [check] SUCCESS: Remote STANDBY file system
/mnt/acfs_rep1 is ONLINE
2020-11-04 21:35:48.588:[acfs_primary]{1:20735:25070} [check] SUCCESS: PRIMARY file system
/mnt/acfs_rep1 is ONLINE
```

The `acfs_primary` 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_rep1  # The replicated ACFS mount point
ATTEMPTS=3                  # Number of attempts to check the remote standby file system
INTERVAL=10                 # Number of seconds between each attempt
```

Use the following command on one node of both the primary and standby clusters to register the ACFS action scripts for monitoring the primary and standby file systems:

```
$ crsctl add resource acfs_primary -type cluster_resource \  
-attr "ACTION_SCRIPT=/u01/oracle/scripts/acfs_primary.scr,CHECK_INTERVAL=60,  
START_DEPENDENCIES='hard(ora.datacl.acfs_repl.acfs) pullup:always(ora.datacl.acfs_repl.acfs)',  
STOP_DEPENDENCIES='hard(ora.datacl.acfs_repl.acfs)',SCRIPT_TIMEOUT=80,OFFLINE_CHECK_INTERVAL=0,  
RESTART_ATTEMPTS=0"
```

Start and check the status of the resources on both the primary and standby clusters:

```
$ crsctl start res acfs_primary  
$ crsctl stat res acfs_primary
```

Refer to [Appendix B](#) for an example `acfs_primary.scr` action script.

acfs_standby

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_oracle.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.

When the standby and remote primary file systems are online, the following is output to the CRS trace file:

```
2020-11-04 21:39:46.752:[acfs_standby]{1:21041:20211} [check] SUCCESS: STANDBY file system  
/mnt/acfs_repl is ONLINE  
2020-11-04 21:39:47.453:[acfs_standby]{1:21041:20211} [check] SUCCESS: Remote PRIMARY file system  
/mnt/acfs_repl is ONLINE
```

When a problem is detected with the primary file system, the following messages are output to the trace file:

```
2020-11-04 21:50:55.967:[acfs_standby]{1:21041:20211} [check] WARNING: Remote PRIMARY file system  
problem detected (attempt 1 of 3)  
2020-11-04 21:51:08.989:[acfs_standby]{1:21041:20211} [check] WARNING: Remote PRIMARY file system  
problem detected (attempt 2 of 3)  
2020-11-04 21:51:22.061:[acfs_standby]{1:21041:20211} [check] WARNING: Remote PRIMARY file system  
problem detected (attempt 3 of 3)  
2020-11-04 21:51:22.061: [acfs_standby]{1:21041:20211} [check] WARNING: Problem with remote PRIMARY  
file system (error: 222)
```

The following parameters use suggested default settings, which should be tested before changing their values.

```
MOUNT_POINT=/mnt/acfs_repl          # This is the replicated ACFS mount point  
ATTEMPTS=8                          # Number of tries to check the remote primary file system  
INTERVAL=10                         # Number of seconds between each attempt
```


Use the following command on one node of both the primary and standby clusters to register the ACFS action scripts for monitoring the standby file system:

```
$ 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.datacl.acfs_repl.acfs,gg_vip_prmy)  
pullup:always(ora.datacl.acfs_repl.acfs,gg_vip_prmy)',OFFLINE_CHECK_INTERVAL=300,RESTART_ATTEMPTS=0  
,INSTANCE_FAILOVER=0"
```

Start and check the status of the resources on both the primary and standby clusters:

```
$ crsctl start res acfs_standby  
$ crsctl stat res acfs_standby
```

Refer to [Appendix C](#) for an example `acfs_standby.scr` action script.

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. The same is true for the `acfs_standby` resource when started on the primary file system cluster.

Refer to [Managing Planned and Unplanned Outages for The GoldenGate Hub](#) for example use cases based on the CRS trace file messages.

F. Test ACFS RAC Node Relocations

It is very important to test planned and unplanned ACFS RAC node relocations and server role transitions before configuring GoldenGate.

To relocate ACFS between primary Oracle RAC nodes:

```
$ srvctl relocate filesystem -diskgroup DATA1 -volume acfs_repl -force
```

After node relocation, verify that the file system is mounted on another node, along with the VIP, NGINX, `sshd_restart`, and the two ACFS resources, `acfs_primary` and `acfs_standby`, using the following example commands.

```
$ srvctl status filesystem -diskgroup DATA1 -volume acfs_repl #Use your diskgroup and volume names  
$ crsctl stat res gg_vip_prmy          # Substitute with your VIP name  
$ crsctl stat res sshd_restart  
$ crsctl stat res acfs_primary  
$ crsctl stat res acfs_standby
```

To issue an ACFS switchover (role reversal) between the primary and standby clusters, run the following example command on the current standby node:

```
# Determine which node is the standby file system node  
$ crsctl stat res gg_vip_prmy          # Substitute with your VIP name  
$ acfsutil repl failover /mnt/acfs_repl # Substitute your ACFS mount point
```

The failover command will do a graceful zero data loss ACFS role reversal if the primary file system is currently available. If the primary file system is not available, an ACFS failover is carried out which may result in file system data loss. This will not be a problem for GoldenGate due to the recovery mechanisms built into the product.

G. ACFS Trace File Management

The ACFS Replication feature can create a large number of trace files located in the CRS trace file directory, which is by default `/u01/app/grid/diag/crs/<node_name>/crs/trace`. It is recommended that you create a crontab task that removes the trace files at regular intervals. Because some of the trace files are generated by the `root` user, the crontab task must be run by the `root` user.

For example, to remove the ACFS generated trace files that are older than 8 hours, you would run the following example crontab entries every 12 hours:

```
0 0,12 * * * /usr/bin/find /u01/app/grid/diag/crs/<node_name>/crs/trace -maxdepth 1 -name
'sdcreate*' -mmin +480 -delete

0 0,12 * * * /usr/bin/find /u01/app/grid/diag/crs/<node_name>/crs/trace -maxdepth 1 -name
'sdapply*' -mmin +480 -delete

0 0,12 * * * /usr/bin/find /u01/app/grid/diag/crs/<node_name>/crs/trace -maxdepth 1 -name
'acfsutil*' -mmin +480 -delete
```

Step 8 - Configure Initial GoldenGate Microservices Deployment

A separate GoldenGate deployment is required for each database release that is being operated against, similar to the Oracle client software and GoldenGate software installations. If both the source and target databases are the same release, only a single deployment is required.

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 due to the fact that the trail files are stored on the same server as the Replicat process.

A GoldenGate Service Manager is created by the first deployment creation. Subsequent deployments are created using the existing Service Manager if they are connecting to the same using the GoldenGate home directory.

There are two ways to create a GoldenGate deployment:

1. Using the GoldenGate Configuration Assistant (`oggca.sh`), a graphical tool for deployment creation
2. Using a response file in silent mode, containing all of the deployment creation parameter values

Both methods of deployment produce the same result, but when creating multiple deployments it is easier to use option #2 with a response file. [Appendix E](#) contains an example response file with parameter changes needed for creating two deployments, one of which is for the different source and target database releases, using GoldenGate 19c. There are no default response files contained in the GoldenGate software installation, so the initial deployment must be created using `oggca.sh`, or the examples provided in [Appendix E](#).

For both deployment creation methods, use the following recommendations.

- Service Manager Options
 - Place the Service Manager deployment home on the ACFS mount point created above in [Step 7](#). This is controlled by the response file parameter `SERVICEMANAGER_DEPLOYMENT_HOME`.
 - Select to 'Integrate with XAG'. This is controlled by the response file parameter `INTEGRATE_SERVICEMANAGER_WITH_XAG`.
 - Set the listening hostname/address to 'localhost' or to ':::1' for IPv6 configurations. This is controlled by response file parameter `HOST_SERVICEMANAGER`. The localhost listening address must be used so that the Service Manager will start on both the primary and standby GoldenGate hub clusters.

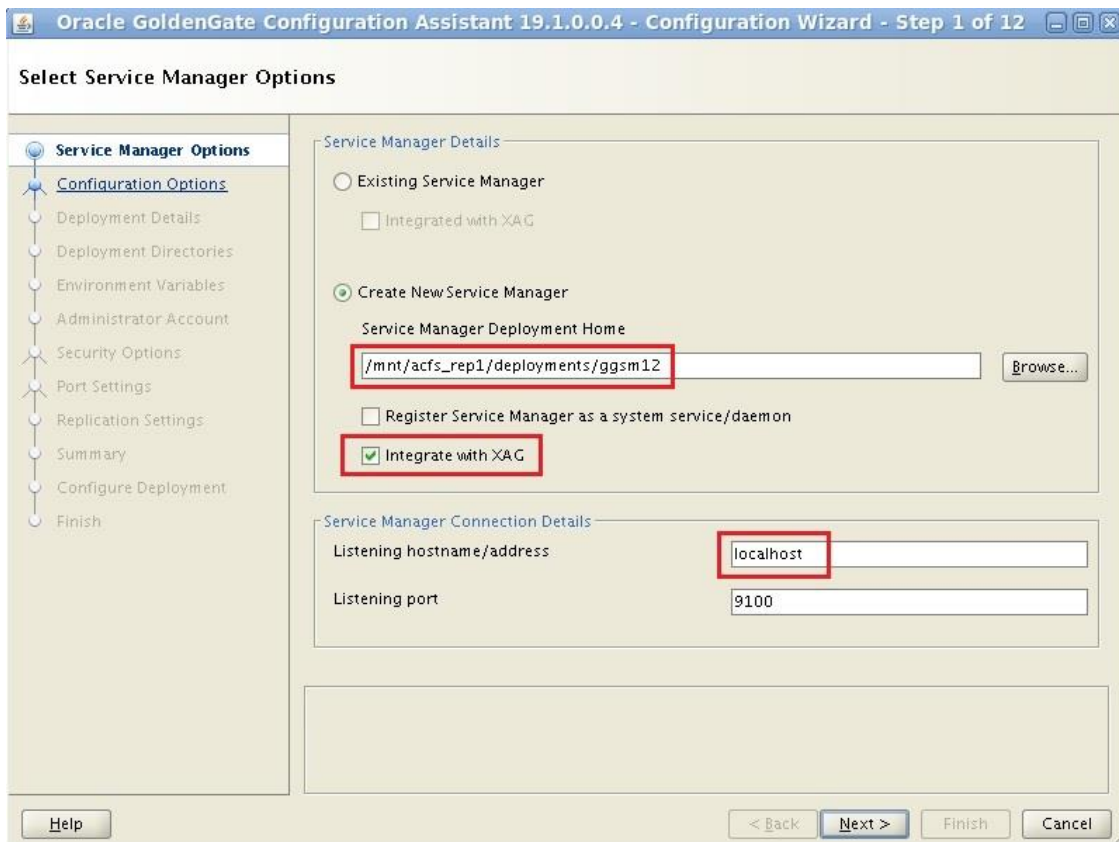


Figure 6: Service Manager deployment creation

- Deployment Directories (Refer to Figure 7)
 - Place the new deployment home on the ACFS mount point created above in [Step 7](#). This is controlled by the response file parameter `OGG_DEPLOYMENT_HOME`.

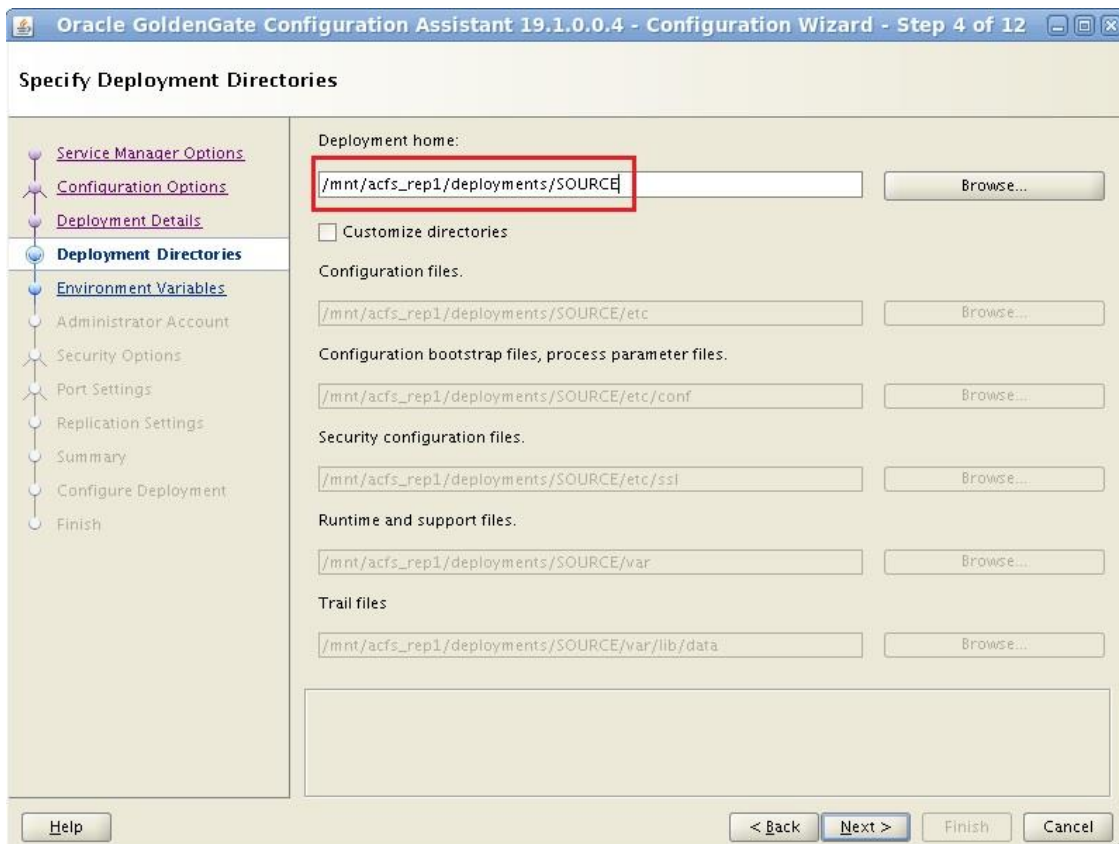


Figure 7: Deployment creation location

- Administrator Account (Refer to Figure 8)
 - Make a note of the GoldenGate administrator account username and password for use by XAG and the REST calls for the deployments. These are controlled by the response file parameters `ADMINISTRATOR_USER` and `ADMINISTRATOR_PASSWORD`.

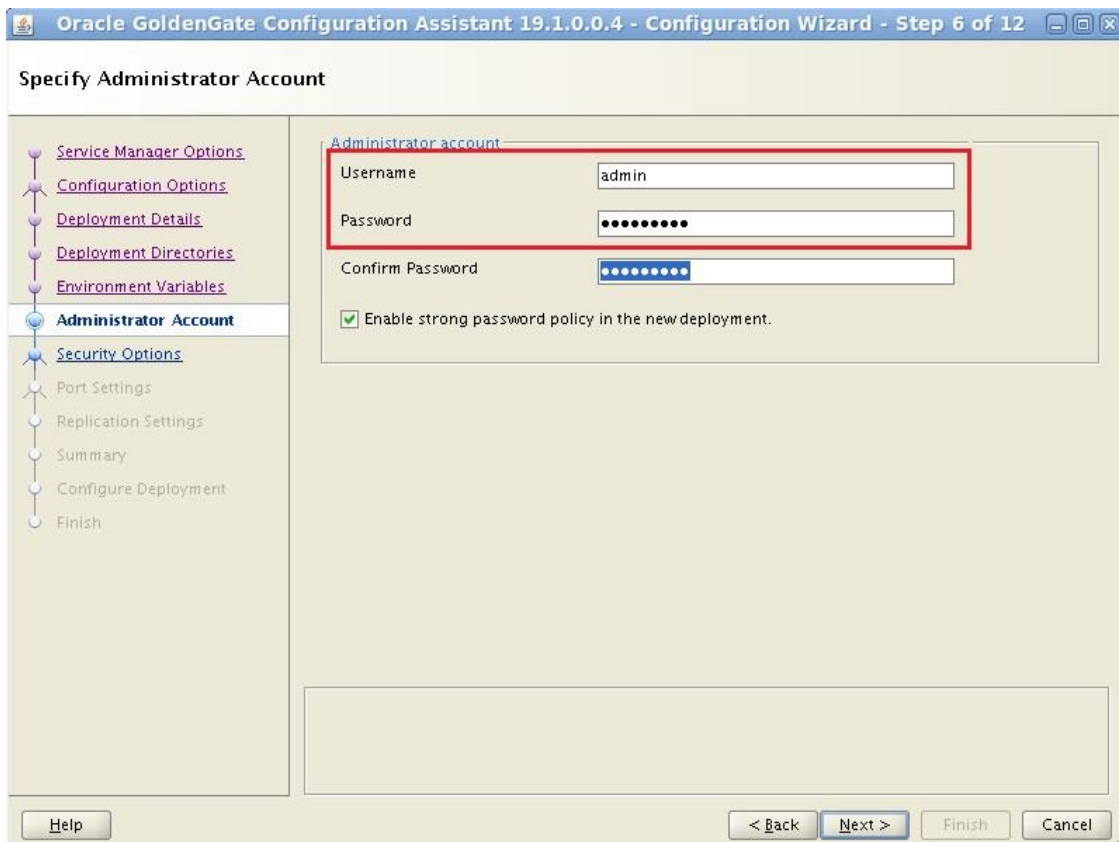


Figure 8: Deployment administrator account

- Security Options
 - Do NOT select to SSL/TLS Security. The NGINX reverse proxy will provide the deployment access security.
- Port Settings (Refer to figure 9)
 - If enabling monitoring and selecting to use Berkley DB (BDB), set the Metrics Server DataStore home' to a location that is NOT on the ACFS replication mount point. Alternatively, use the Lightning Memory-Mapped Database (LMDB) for the DataStore type. These are controlled by the response file parameters `PMSRVR_DATASTORE_TYPE` and `PMSRVR_DATASTORE_HOME`.

Oracle GoldenGate Configuration Assistant 19.1.0.0.4 - Configuration Wizard - Step 8 of 12

Specify Port Settings

- Service Manager Options
- Configuration Options
- Deployment Details
- Deployment Directories
- Environment Variables
- Administrator Account
- Security Options
- Port Settings**
- Replication Settings
- Summary
- Configure Deployment
- Finish

Service Manager Details

Listening hostname/address: 127.0.0.1 Listening port: 9100

Servers

☒ Enable Administration Server Administration Server port: 9101

☐ Enable Distribution Server Distribution Server port: 9102

☐ Enable Receiver Server Receiver Server port: 9103

Monitoring

☒ Enable Monitoring ☐ XAG Critical

Metrics Server port: 9104

Metrics Server UDP port (data): 9105

Metrics Server DataStore type: BDB

Metrics Server DataStore home: /oracle/goldengate/gg19_db12_MS/dirbdb

Help < Back Next > Finish Cancel

Figure 9: Metric server datastore parameters

If you are using the GoldenGate Configuration Assistant (`oggca.sh`) to create the initial deployment, it is recommended that you save the response file to be used as the template for future deployment creations. Figure 10 shows the final configuration assistant screen where the response file can be saved.

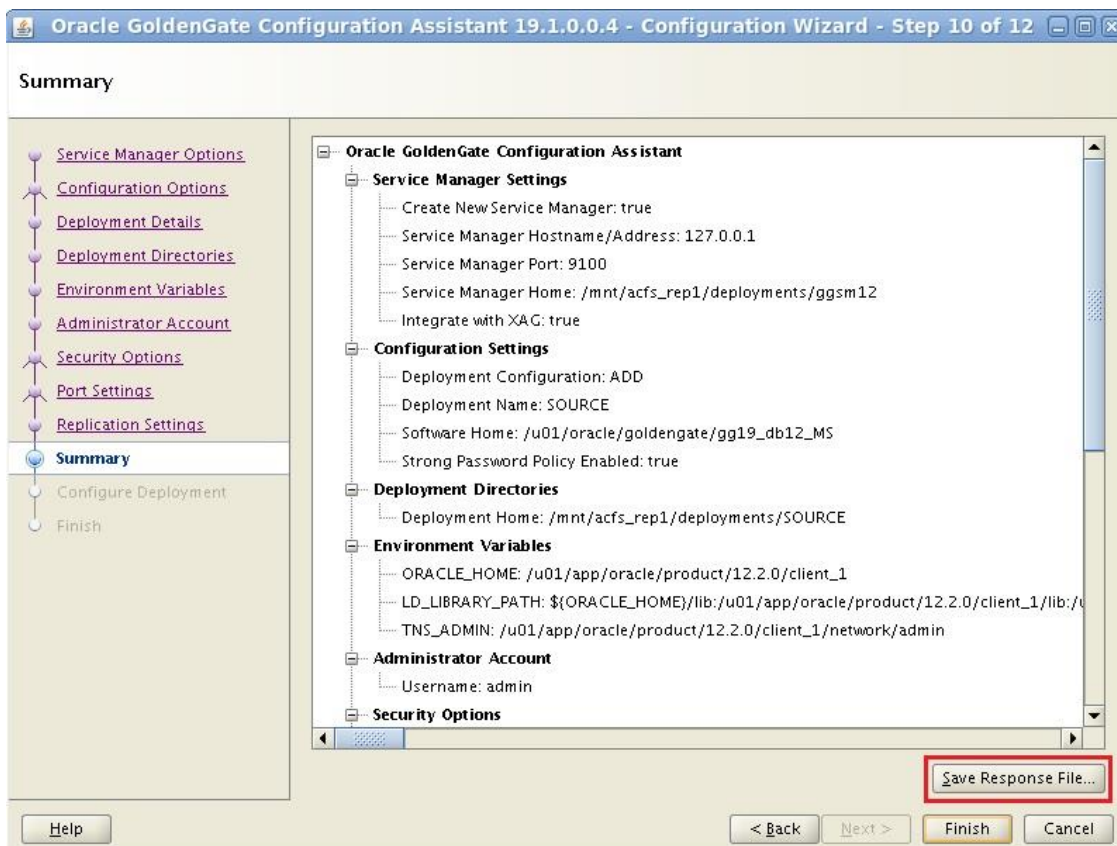


Figure 10: Saving the deployment creation response file

If you are using the example response file in [Appendix E](#) to set the correct parameters, create the initial deployment using the following command.

```
$ cd $GG_HOME/bin
$ ./oggca.sh -silent -responseFile /u01/oracle/target_deployment.rsp
```

When complete the following message will be displayed:

```
Linux, x64, 64bit (optimized) on Jan 01 2020 14:38:11
Operating system character set identified as UTF-8.
Stopping...STOPPED.
Successfully Setup Software.
```

If the source and target databases are different Oracle releases, for example 12g Release 2 and 19c, an additional deployment must be created in a later step (see [Step 9 - Create Additional GoldenGate Deployments](#)).

Throughout this technical brief, two GoldenGate deployments will be referenced, named SOURCE and TARGET.

Step 9 - Create Additional GoldenGate Deployments

If additional GoldenGate deployments are required, due to the source and target databases being of different versions, repeat the deployment configuration steps provided above in [Step 8 - Configure Initial GoldenGate Microservices Deployment](#). Due to a current limitation in the way XAG starts GoldenGate deployments running in different OGG_HOME directories, each deployment with a different OGG_HOME directory must be created with a new Service Manager. For example, if creating two deployments to support both an Oracle 12c and Oracle 19c database, using separate OGG_HOME directories, two Service Managers would be required.

NOTE: Make sure all Service Manager and GoldenGate Servers created (admin server, performance metric server etc.) are using different network port numbers.

Refer to [Appendix E](#) for an example response file for creating the second deployment.

Step 10 - Register GoldenGate Deployments with Grid Infrastructure Standalone Agent

Registering the deployments with the Grid Infrastructure standalone (XAG) agent will create the correct start and stop dependencies between the GoldenGate service managers, the VIP and the ACFS file system service.

If you are using existing GoldenGate deployments that were not created as part of the configuration steps provided in this paper, and the deployments were managed by the operating system daemon (systemd) they must be removed from systemd before they can be registered with XAG.

Use the following commands to check if the deployment is managed by systemd, and then remove it from systemd.

```
# systemctl list-unit-files|grep -i goldengate
# systemctl stop OracleGoldenGate
# systemctl disable OracleGoldenGate
```

To register a deployment with XAG use the following command format, run as the `root` user:

```
agctl add goldengate <instance_name>
  --gg_home <GoldenGate_Home>
  --service_manager
  --config_home <GoldenGate_SvcMgr_Config>
  --var_home <GoldenGate_SvcMgr_VarDir>
  --port <port number>
  --adminuser <OGG admin user>
  --user <GG instance user>
  --group <GG instance group>
  --filesystems <CRS_resource_name>
  --filesystem_verify <yes|no>
  --filesystems_always yes
  --attributes 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.
- `--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 RAC 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 hub where ACFS is mounted in either read-write or read-only mode. To determine which node of the cluster the file system is mounted on, run the following command:

```
$ crsctl stat res acfs_standby

NAME=acfs_standby
TYPE=cluster_resource
TARGET=ONLINE
STATE=ONLINE on pnode01
```

Below is an example of registering GoldenGate with XAG (as root):

```
# agctl add goldengate TARGET \
--gg_home /u01/oracle/goldengate/gg19_db19_MS \
--service_manager \
--config_home /mnt/acfs_rep1/deployments/ggsm19/etc/conf \
--var_home /mnt/acfs_rep1/deployments/ggsm19/var \
--port 9100 \
--adminuser admin \
--user oracle \
--group oinstall \
--filesystems acfs_primary \
--filesystem_verify yes \
--filesystems_always yes \
--attribute TARGET_DEFAULT=online
```

Below are some example `agctl` commands that are used to manage the GoldenGate deployment with XAG.

To check the status of GoldenGate:

```
% agctl status goldengate

Goldengate instance 'TARGET' is running on pnode01
```

To start the GoldenGate deployment service manager, which in turn will start all deployments, and GoldenGate processes:

```
% agctl start goldengate TARGET --node pnode02
```

To stop the GoldenGate deployments managed by the service manager:

```
% agctl stop goldengate TARGET
```

To manually relocate the GoldenGate deployment to another node:

```
% agctl relocate goldengate TARGET --node pnode01
```

Note: The GoldenGate resource *MUST* be running before relocating it.

To view the configuration parameters for the GoldenGate instance:

```
% agctl config goldengate TARGET

Instance name: GG_MS01
Application GoldenGate location is: /u01/oracle/goldengate/gg19_db19_MS
```

```
Goldengate MicroServices Architecture environment : yes
Goldengate Service Manager configuration directory : /mnt/acfs_repl/deployments/ggsm19/etc/conf
Goldengate Service Manager var directory : /mnt/acfs_repl/deployments/ggsm19/var
Service Manager Port : 9100
Goldengate Administration User : admin
File System resources needed: acfs_primary
CRS additional attributes set: TARGET_DEFAULT=online
```

To view more detailed clusterware configuration information:

```
% crsctl stat res -w "NAME = xag.TARGET.goldengate" -p
```

To delete the GoldenGate instance, stop the processes and then delete the resource:

```
% actl stop goldengate TARGET
```

Delete of the GoldenGate instance must be carried out as root:

```
# agctl remove goldengate TARGET
```

NOTE: Before proceeding to the next step, make sure the GoldenGate deployments have been started with XAG.

Further information on the Oracle Grid Infrastructure Bundled Agent:

<http://www.oracle.com/technetwork/database/database-technologies/clusterware/downloads/xag-agents-downloads-3636484.html>

Step 11 - Configure NGINX Reverse Proxy

A separate reverse proxy configuration is required for each OGG_HOME Service Manager on both the primary and standby clusters. For example, the first Service Manager can be accessed with port 443, and the second Service Manager can be accessed with port 444.

1. Create a server certificate for NGINX

A certificate is required for NGINX to authenticate client requests using HTTPS. Contact your systems administrator to follow your corporate standards to create or obtain the server certificate.

Once obtained, the certificate should be copied to the location `/etc/nginx/ogg.pem`.

The new certificate will be read by NGINX later when the configuration is reloaded.

2. Create the NGINX configuration files

Make sure the `OGG_HOME` environment variable is set to the home directory for the GoldenGate software running the Service Manager.

NOTE: The GoldenGate Service Manager MUST be running before creating the NGINX configuration files using the following command.

```
export OGG_HOME=/u01/app/oracle/goldengate/gg19_db19_MS
$ $OGG_HOME/lib/utl/reverseproxy/ReverseProxySettings -u <username> -P <password> -p 443 -o
ogg_db19.conf http://localhost:9100
```

The user name and password specified by the `-u` and `-P` parameters should mirror the GoldenGate administrator account specified with the initial deployment creation (in the `ADMINISTRATOR_USER` and `ADMINISTRATOR_PASSWORD` response file parameters). The reverse proxy port number specified by the `-p` parameter must be an unused port number that is different for each GoldenGate

deployment running its own service manager. Lastly, the http port number (9100) should also match the Service Manager port number specified in the deployment creation (in the `PORT_SERVICEMANAGER` response file parameter).

Repeat this step for each of the GoldenGate deployments created with their own service manager. Example command for the second deployment:

```
export OGG_HOME=/u01/app/oracle/goldengate/gg19_db12_MS
export PATH=$PATH:$OGG_HOME/bin

$ $OGG_HOME/lib/utl/reverseproxy/ReverseProxySettings -u <username> -P <password> -p 444 -o
ogg_db12.conf http://localhost:9110
```

NOTE: This command needs to be re-run every time the GoldenGate deployment configurations changes, for example when adding or removing a deployment from the same OGG_HOME Service Manager.

3. Install the NGINX configuration files

```
$ sudo mv ogg_db19.conf /etc/nginx/conf.d/
```

Repeat this step to copy the second deployment configuration file, if created in the previous step.

4. Test the new NGINX configuration

```
$ sudo nginx -t

nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

5. Reload NGINX and the new configuration

If NGINX is not currently running, start it and load the new configuration:

```
$ sudo nginx
$ sudo nginx -s reload
```

6. Test GoldenGate Microservices connectivity

A simple test is to query the health of the deployments using the following command.

```
$ curl -s -K access.cfg https://localhost:<port#>/services/v2/config/health -XGET | python -m
json.tool
```

NOTE: The port number is only required if not using the standard HTTPS port of 443.

Sample output:

```
{
  "$schema": "api:standardResponse",
  "links": [
    {
      "href": "https://localhost/services/v2/config/health",
      "mediaType": "application/json",
      "rel": "canonical"
    },
    {
```

```

...
  "response": {
    "$schema": "ogg:health",
    "criticalResources": [
      {
        "deploymentName": "source",
        "healthy": true,
        "name": "adminsrvr",
        "status": "running",
        "type": "service"
      },
      {
        "deploymentName": "target",
        "healthy": true,
        "name": "adminsrvr",
        "status": "running",
        "type": "service"
      },
    ],
    "deploymentName": "ServiceManager",
    "healthy": true,
    "serviceName": "ServiceManager",
    "started": "2019-03-28T21:52:42.835Z"
  }
}

```

7. Distribute the GoldenGate NGINX configuration files to all environments

Repeat steps 3 to 5 on every ACFS file system primary and standby RAC cluster, copying the configuration files created in [step 2](#).

Testing the GoldenGate Microservices connectivity can only be done on the current node running the deployments.

8. Create NGINX Cluster Resource

The running status of the NGINX reverse proxy needs to be controlled by Oracle CRS so that it can automatically be started before the GoldenGate deployments are started.

As the root user, use the following example command to create a CRS resource to manage NGINX.

```

$ crsctl add resource nginx -type generic_application -attr
"ACL='owner:root:rw,prgp:root:rw,other::r--,group:oinstall:r-
x,user:oracle:rw',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(gg_vip_prmy) pullup:always(gg_vip_prmy)
attraction(gg_vip_prmy)',STOP_DEPENDENCIES='hard(gg_vip_prmy)' "

```

NOTE: The example above is for Linux 7 and above. For Linux 6, replace `/sbin/systemctl` with `/etc/rc.d/init.d/nginx` `start|stop|status`.

9. Modify the GoldenGate XAG instances for NGINX

Once the NGINX CRS resource is created, the GoldenGate XAG resources need to be altered so that the NGINX reverse proxy server is started when the GoldenGate deployments are started.

As the oracle user, modify the XAG resource using the following example command.

```
$ agctl modify goldengate SOURCE --filesystems acfs_primary,nginx
$ agctl modify goldengate TARGET --filesystems acfs_primary,nginx
```

GOLDENGATE CONFIGURATION

This section describes the GoldenGate configuration steps that must be done on the hub.

Creating the Oracle Net Services Client Configuration Files

Using the role based service names that were created in earlier steps ([Create Database Role Based Service](#)), create a source and target database TNS alias in the tnsnames.ora file in the respective GoldenGate deployment TNS_ADMIN directory. The location of the deployment TNS_ADMIN directory is can be determined using the following command:

```
$ curl -s -K access.cfg https://localhost:<port#>/services/v2/deployments/<deployment_name> -XGET |
python -m json.tool
```

Example:

```
curl -s -K access.cfg -k https://localhost:443/services/v2/deployments/TARGET -XGET | python -m
json.tool|grep -A1 TNS_ADMIN

      "name": "TNS ADMIN",
      "value": "/u01/app/oracle/product/19.0.0.0/client_1/network/admin"
```

The example sqlnet.ora file below contains recommended parameters for GoldenGate.

```
DEFAULT_SDU_SIZE = 2097152

# Uncomment, if using Oracle Net encryption:
#SQLNET.ENCRYPTION_CLIENT = required
#SQLNET.ENCRYPTION_TYPES_CLIENT= (AES256)
```

For GoldenGate Replicat and Extract processes to detect a network outage between the hub and target database host in a timely fashion, so that they will reconnect to an alternative database instance according to the TNS alias definition, it is recommended to set the SQLNET.RCV_TIMEOUT TNS alias parameter.

The value of SQLNET.RECV_TIMEOUT should be set to 30 (seconds) for a TNS alias used by Extract, and 120 (seconds) for a Replicat TNS alias.

Use the following example to create a GoldenGate Extract TNS alias for a RAC database that is configured with Oracle Data Guard:

```
ggsource =
(DESCRIPTION_LIST=
  (LOAD_BALANCE=off) (FAILOVER=on) (CONNECT_TIMEOUT=3) (RETRY_COUNT=2)
  (DESCRIPTION =
    (RECV_TIMEOUT=30)
    (ADDRESS_LIST = (LOAD_BALANCE=off) (FAILOVER=on) (CONNECT_TIMEOUT=3) (RETRY_COUNT=3)
      (ADDRESS = (PROTOCOL = TCP) (HOST = pnode01) (PORT = 1521))
      (ADDRESS = (PROTOCOL = TCP) (HOST = pnode02) (PORT = 1521))
    )
    (CONNECT_DATA = (SERVICE_NAME = oggsource.example.com))
  )
  (DESCRIPTION =
    (RECV_TIMEOUT=30)
    (ADDRESS_LIST = (LOAD_BALANCE=off) (FAILOVER=on) (CONNECT_TIMEOUT=3) (RETRY_COUNT=3)
      (ADDRESS = (PROTOCOL = TCP) (HOST = snode01) (PORT = 1521))
```

```

        (ADDRESS = (PROTOCOL = TCP) (HOST = snode02) (PORT = 1521))
    )
    (CONNECT_DATA = (SERVICE_NAME = oggsource.example.com))
)
)

```

NOTE: If using multiple GoldenGate deployments, make sure the TNS alias is created in the correct deployment TNS_ADMIN tnsnames.ora file.

GoldenGate Extract Process Recommendations

GoldenGate Integrated Extract is a requirement when extracting data from a source database that is configured with Oracle Data Guard, in order to continue extracting through an Oracle Data Guard role transition.

Oracle Data Guard Data Loss Failover Considerations

For Data Guard configurations, the following parameters control how the integrated Extract process operates following a Data Guard failover with the source database:

TRANLOGOPTIONS HANDLEDLFAILOVER

This parameter prevents Extract from extracting redo data 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.

After a Data Guard failover, if the old primary (the new standby) database is not available, Extract will continue extracting data from the new primary database until the value of `DLFAILOVER_TIMEOUT` seconds has past (defaults to 300 seconds). Extract will report the following message into the report file indicating the timeout period has begun:

```

2020-05-15 14:32:59 INFO    OGG-25052  The primary database has undergone a role transition.
Temporarily suspend the HANDLEDLFAILOVER behavior for 300 seconds to allow time for standby
database reinstatement. Note that for the duration of this timeout period, Extract will not
throttle trail generation. Therefore, there is a potential for data divergence between the Oracle
GoldenGate target database and the reinstating Oracle Data Guard standby database.

```

If the new standby database has not become available by the time the timeout expires, Extract will abend with an error similar to the following:

```

2020-05-15 14:38:00 INFO    OGG-25053  Timeout waiting for 300 seconds for standby database
reinstatement. Now enforcing HANDLEDLFAILOVER.
2020-05-15 14:38:30 ERROR   OGG-06219  Unable to extract data from the Logmining server
OGG$CAP_EXT1.

```

At this point, if the number of automatic retries at start Extract have reached the auto-restart task limit (refer to [Create Autostart Profiles](#)), the Extract will need to be manually restarted when the old primary database is reinstated as standby database.

If the old primary database will be unavailable for an extended period of time or completely gone, then in order for GoldenGate replication to continue, you must remove the `HANDLEDLFAILOVER` parameter from the Extract parameter file. Extract will no longer wait 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 GoldenGate target database. This will be resolved once the standby database is up to date. At which time the `HANDLEDLFAILOVER` parameter should be added back into the Extract parameter file.

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 `HANDEDLFAILOVER` 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.

TRANLOGOPTIONS DLFAILOVER_TIMEOUT

This parameter provides a configurable timeout in seconds to allow for standby database reinstatement post-failover. It is used in conjunction with `HANDEDLFAILOVER` to allow Integrated Extract to start up immediately after a role transition. At the end of the timeout period, if the standby database is still not available, then Extract will terminate. The default timeout is 300 seconds (5 minutes), and is suitable for most configurations.

TRANLOGOPTIONS FAILOVERTARGETDESTID

Oracle recommends configuring Fast-Start Failover (FSFO) so that the Oracle Data Guard broker can automatically fail over to a previously chosen standby database in the event of a loss of the primary database. Without FSFO, failing over to the standby database is a manual process.

If FSFO is disabled, you must specify the `FAILOVERTARGETDESTID` Extract parameter.

This parameter identifies which Oracle Data Guard standby database the 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`, the `archive_log_dest` database initialization parameter is used. Replace `n` with the correct archive log destination identifier.

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="ggs2d", ASYNC NOAFFIR
                                     M delay=0 optional compression
                                     =disable max_failure=0 max_con
                                     nections=1 reopen=300 db_uniqu
                                     e_name="GGS2D" net_timeout=30,
                                     valid_for=(online_logfile,all
                                     _roles)
```

In this example, the Extract parameter would be set to the following:

```
TRANLOGOPTIONS FAILOVERTARGETDESTID 2
```

Here's an example Extract creation command using the required parameters:

```
curl -s -K access.cfg https://gghub.example.com/SOURCE/adminsrvr/services/v2/extracts/EXT1 -X POST
--data '{"config":["Extract EXT1", "ExtTrail
/mnt/acfs_repl/deployments/SOURCE/var/lib/data/aa","UseridAlias source_db DOMAIN
goldengate","ENCRYPTTRAIL AES256","TRANLOGOPTIONS PERFORMANCEPROFILE HIGH", "TRANLOGOPTIONS
_readaheadcount 64", "TRANLOGOPTIONS HANDLEDLFAILOVER", "TRANLOGOPTIONS FAILOVERTARGETDESTID
2","REPORTCOUNT EVERY 15 MINUTES, RATE","STATOPTIONS REPORTFETCH", "DDL EXCLUDE ALL","Table
soesmall.*;"],"source":{"tranlogs":"integrated"},
"credentials":{"alias":"source_db","domain":"goldengate"},"begin":"now","targets":[{"path":"/mnt/ac
fs_repl/deployments/SOURCE/var/lib/data","name":"aa","sizeMB":500}], "registration":"default"}'|
python -m json.tool
```

NOTE: The `_readaheadcount` parameter is required after the patch for bug 28849751 is applied to the source on-premises database.

NOTE: If you are extracting multiple schemas, use multiple Table <Source PDB>.<Schema Name>.; parameters.*

To confirm the Extract process was created, us the following example command.

```
curl -s -K access.cfg https://gghub.example.com/SOURCE/adminsrvr/services/v2/extracts/EXT1 -X GET |
python -m json.tool
```

Sample output:

```
    "config": [
        "Extract EXT1",
        "ExtTrail /mnt/acfs_repl/deployments/SOURCE/var/lib/data/aa",
        "UseridAlias source_db DOMAIN goldengate",
        "ENCRYPTTRAIL AES256",
        "TRANLOGOPTIONS PERFORMANCEPROFILE HIGH",
        "TRANLOGOPTIONS _readaheadcount 64",
        "TRANLOGOPTIONS HANDLEDLFAILOVER",
        "TRANLOGOPTIONS FAILOVERTARGETDESTID 2",
        "REPORTCOUNT EVERY 15 MINUTES, RATE",
        "STATOPTIONS REPORTFETCH",
        "DDL EXCLUDE ALL",
        "Table soesmall.*;"
    ],
    "credentials": {
        "alias": "source_db",
        "domain": "goldengate"
    },
    "encryptionProfile": "LocalWallet",
    "registration": {
        "csn": 1109250433
    },
    "source": {
        "tranlogs": "integrated"
    },
    "status": "stopped"
```

GoldenGate Replicat Process Recommendations

Oracle generally recommends using non-integrated parallel Replicat which offers better apply performance for most workloads when the network latency between the hub and the target database is low ($\leq 4\text{ms}$).

Under the following circumstances, integrated parallel Replicat should be used.

- Using any of the following Replicat features:
 - Data Manipulation Language (DML) apply handler or error handler.
 - Automatic Conflict Detection and Resolution (CDR).
 - Procedural replication.
- There is a higher network latency between the Oracle hub and target database (>4ms).

The best apply performance can be achieved when the network latency between the hub and the target database is as low as possible. The following configuration is recommended for the remote Replicat running on the Oracle hub.

- `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.

More information on GoldenGate parallel Replicat can be found in the Using GoldenGate for Oracle database guide at

<https://docs.oracle.com/en/middleware/goldengate/core/19.1/oracle-db/index.html>

GoldenGate Distribution Path Recommendations

If there are any GoldenGate distribution paths sending trail files to the hub, and a wide area network virtual IP address that can relocate between different servers is not being used, after a role transition of the hub the paths will need to be altered to send the trail files to the new primary hub server. This can be done using the following example REST call:

```
curl -s -K src_access.cfg https://<source host>/<source deployment
name>/distsrvr/services/v2/sources/<distribution path name> -X PATCH --data
'{"target":{"uri":"ogg://scao04client07-vip1:9103/services/v2/targets?trail=dd"}}' | python -m
json.tool
```

If desired, the changing of the source distribution path target address could be automated after a hub role transition using the sample shell script shown in [Appendix D](#), which is called by the `acfs_standby` CRS action script (see [Appendix C](#)) when a file system switchover/failover occurs.

The source distribution paths must be configured to restart automatically after it has failed, so that in the event of the target GoldenGate deployment relocating 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://ggsources.example.com/GG01/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://ggsources.example.com/GG01/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
    }
  },
```

Create Autostart Profiles

Autostart and autorestart profiles are needed to automatically start the GoldenGate processes when the deployment is started by the Service Manager. By default, newly created GoldenGate processes are not part of an active profile to automatically start them when the deployment starts.

Use the following steps to configure an autostart profile and assign it to GoldenGate processes.

1. Create the following autostart and autorestart profiles for both the source and target deployments.

```
# SOURCE deployment
$ curl -s -K access.cfg https://gghub.example.com/SOURCE/adminsrvr/services/v2/config/types/ogg:managedProcessSettings/values/ogg:managedProcessSettings:AUTOSOURCE01 -XPOST --data '{"autoStart": {"enabled": true, "delay": 10}, "autoRestart": {"delay": 30, "disableOnFailure": true, "enabled": true, "onSuccess": false, "retries": 5, "window": 1200}}' | python -m json.tool

# TARGET deployment
$ curl -s -K access.cfg https://gghub.example.com/TARGET/adminsrvr/services/v2/config/types/ogg:managedProcessSettings/values/ogg:managedProcessSettings:AUTOTARGET01 -XPOST --data '{"autoStart": {"enabled": true, "delay": 10}, "autoRestart": {"delay": 30, "disableOnFailure": true, "enabled": true, "onSuccess": false, "retries": 5, "window": 1200}}' | python -m json.tool
```

2. Assign the autostart profile to the Extract and Replicat processes.

```
$ curl -s -K access.cfg https://gghub.example.com/SOURCE/adminsrvr/services/v2/extracts/EXT1 -X PATCH --data '{"managedProcessSettings": "AUTOSOURCE01"}' | python -m json.tool

$ curl -s -K access.cfg https://gghub.example.com/TARGET/adminsrvr/services/v2/replicats/REP1 -X PATCH --data '{"managedProcessSettings": "AUTOTARGET01"}' | python -m json.tool
```

Confirm that the GoldenGate processes are configured with the new profile (partial output shown).

```
$ curl -s -K access.cfg https://gghub.example.com/SOURCE/adminsrvr/services/v2/extracts/EXT1 -X GET | python -m json.tool | grep managedProcessSettings

  "managedProcessSettings": "AUTOSOURCE01",

$ curl -s -K access.cfg https://gghub.example.com/TARGET/adminsrvr/services/v2/replicats/REP1 -X GET | python -m json.tool | grep managedProcessSettings

  "managedProcessSettings": "AUTOTARGET01",
```

Monitoring GoldenGate Processes

End-to-end GoldenGate replication latency should be monitored using the automatic heartbeat functionality. The automatic heartbeats are sent from each source database into the replication streams, by updating the records in a heartbeat seed table and a heartbeat table, and constructing a heartbeat history table. Each of the replication processes in the replication path process these heartbeat records and update the information in them. These heartbeat records are inserted or updated into the heartbeat table at the target databases.

The heartbeat objects need to be installed on both the source and target databases, using the correct database credential previously created. If the database is a Multitenant database, the heartbeat table must be created in the PDB.

Use the following example commands to create and confirm the heartbeat object creation.

```
$ curl -s -K access.cfg
https://ggub.example.com/SOURCE/adminsrvr/services/v2/connections/goldengate.source_db/tables/heart
beat -X POST --data '{"frequency":5}' | python -m json.tool

$ curl -s -K access.cfg -k
https://gghub.example.com/SOURCE/adminsrvr/services/v2/connections/goldengate.source_db/tables/hear
tbeat -XGET | python -m json.tool
```

Basic GoldenGate monitoring of process status and latency can be carried out with the following example commands.

Monitoring Replication Latency

Query the heartbeat table in the target deployment using the following command. This shows the latency from source database changes to target database replication.

```
$ curl -s -K access.cfg
https://gghub.example.com/TARGET/adminsrvr/services/v2/connections/goldengate.target_db/tables/hear
tbeat/REP1 -XGET | python -m json.tool | grep -e ageSeconds -e lagSeconds

# Sample output:
  "response": {
    "$schema": "ogg:processHeartbeat",
    "heartbeats": [
      {
        "ageSeconds": 2.02,
        "lagSeconds": 0.63,
        "path": "EXT1 ==> REP1",
        "source": "GGS2",
        "target": "GGT"
      }
    ]
  }
```

Viewing Extract and Replicat Process Status

GoldenGate process status can be viewed with the following command.

```
$ curl -s -K access.cfg
https://gghub.example.com/SOURCE/adminsrvr/services/v2/replicats/EXT/info/status -X GET | python -m
json.tool

# Sample partial output:
  "response": {
    "$schema": "ogg:extractStatus",
    "lag": 1,
```

```

    "lastStarted": "2020-08-06T06:01:34.605Z",
    "position": "0.1118127854",
    "processId": 98612,
    "sinceLagReported": 6,
    "status": "running"
  }

```

Viewing Extract and Replicat Report Files

When an Extract or Replicat process abends, the process report file can be viewed using the following command.

```

curl -s -K access.cfg
https://gghub.example.net/SOURCE/adminsrvr/services/v2/extracts/EXT1/info/reports/EXT1.rpt -X GET |
python -m json.tool

# Sample partial output:

"response": {
  "$schema": "ogg:report",
  "lines": [
    "",
    "*****",
    "          Oracle GoldenGate Delivery for Oracle",
    "    Version 19.1.0.0.4 OGGCORE_19.1.0.0.0_PLATFORMS_191017.1054_FBO",
    "    Linux, x64, 64bit (optimized), Oracle 19c on Oct 17 2019 21:17:25",
    " ",
    "Copyright (C) 1995, 2019, Oracle and/or its affiliates. All rights reserved.",
    "",
    "          Starting at 2020-08-05 23:32:26",
    "*****",
    "",
    "Operating System Version:",
    "Linux",
    "Version #2 SMP Wed May 8 22:25:03 PDT 2019, Release 4.1.12-124.26.12.el7uek.x86_64",

```

It is recommended that you use the GoldenGate Management Pack to monitor GoldenGate processes. Management Pack for GoldenGate includes three complementary products: the GoldenGate Enterprise Manager Plug-in for monitoring and managing GoldenGate in the context of your entire enterprise; GoldenGate Monitor for focused monitoring and managing of GoldenGate solutions only; and GoldenGate Director for managing and monitoring legacy GoldenGate components. Together they empower the new and seasoned GoldenGate users alike with enterprise level monitoring, alerting, and management of GoldenGate solutions.

MANAGING PLANNED AND UNPLANNED OUTAGES FOR THE 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.

Planned Outages

When there is a requirement to perform planned maintenance on the GoldenGate hub, 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.

Primary GoldenGate Hub

Before maintenance can be carried out on the primary hub cluster, do the following:

1. Perform an ACFS role reversal so that the standby becomes the new primary. With both primary and standby file systems online, the `repl failover` command ensures that all outstanding primary file system changes are transferred and applied to the standby before the role reversal completes.

On the current standby file system server, run the following command:

```
$ /sbin/acfsutil repl failover /mnt/acfs_repl # Specify the correct mount point
$ /sbin/acfsutil repl info -c -v /mnt/acfs_repl # Run on both primary and standby servers to
# confirm ACFS replication role reversal
$ agctl start goldengate SOURCE # Specify the GoldenGate instance name
```

2. If you want to disable the CRS trace file messages on new standby cluster, stop and disable the `acfs_standby` resource on the new standby cluster.

Run the following command on the current standby hub cluster to disable `acfs_standby` resource.

```
$ crsctl stop res acfs_standby
$ crsctl modify resource acfs_standby -attr "ENABLED=0"
```

After the maintenance is complete, reverse the steps to revert the configuration to its state before the maintenance:

1. Enable and start the `acfs_standby` resource on the standby cluster

```
$ crsctl modify resource acfs_standby -attr "ENABLED=1"
$ crsctl start res acfs_standby
```

2. If desired, switchover the ACFS file system back to the original ACFS replication direction.

On the current standby file system server, run the following command:

```
$ /sbin/acfsutil repl failover /mnt/acfs_repl # Specify the correct mount point
$ /sbin/acfsutil repl info -c -v /mnt/acfs_repl # Run on both primary and standby servers to
# confirm ACFS replication role reversal
$ agctl start goldengate SOURCE # Specify the GoldenGate instance name
```

Standby GoldenGate Hub

Before maintenance can be carried out on the standby hub cluster, stop and disable the `acfs_standby` resource on the standby cluster so it won't continue to write warnings in the CRS trace file:

```
$ crsctl stop res acfs_standby
$ crsctl modify resource acfs_standby -attr "ENABLED=0"
```

After the maintenance is complete, reverse the steps to revert the configuration back to the way it was before the maintenance. Enable and start the `acfs_standby` resource on the standby cluster:

```
$ crsctl modify resource acfs_standby -attr "ENABLED=1"
$ crsctl start res acfs_standby
```

Unplanned Outages of the GoldenGate Hub

When an unplanned outage occurs on either the primary or standby GoldenGate hub clusters, there are some instructions to ensure the continuous operation of GoldenGate. Use the following hub failure use cases to guide you in the event of an unplanned outage of the primary and standby hub cluster.

Use case #1 – Standby Hub Failure or Primary Hub Cannot Communicate with the Standby Hub

If the primary hub cannot communicate with the standby hub, the following messages will be output into the primary CRS trace file (`crsd_scriptagent_oracle.trc`) on the active cluster node:

```
2020-11-16 15:15:01.506:[acfs_primary]{1:40360:26097} [check] WARNING: STANDBY not accessible
(attempt 1 of 3)
2020-11-16 15:15:14.535:[acfs_primary]{1:40360:26097} [check] WARNING: STANDBY not accessible
(attempt 2 of 3)
2020-11-16 15:15:27.611:[acfs_primary]{1:40360:26097} [check] WARNING: STANDBY not accessible
(attempt 3 of 3)
2020-11-16 15:15:27.611:[acfs primary]{1:40360:26097} [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 GoldenGate will continue to function unimpeded.

Use the following action plan with this scenario.

- Check the standby file system, using the command `'/sbin/acfsutil repl util verifystandby /mnt/acfs_repl -v'` to determine why the standby hub is inaccessible. Refer to [Appendix F](#) for common AFS Replication errors and their solutions.
- After fixing the cause of the communication errors, the standby will automatically 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:

```
2020-11-16 15:28:56.097: [acfs_primary]{1:40360:26097} [check] SUCCESS: STANDBY file system
/mnt/acfs_repl is ONLINE
```

Use case #2 – Primary Hub Failure or Standby Hub Cannot Communicate with the Primary Hub

If the standby hub cannot communicate with the primary hub, the the following messages will be output into the standby CRS trace file (`crsd_scriptagent_oracle.trc`) on the active cluster node:

```
2020-11-16 15:31:42.198:[acfs_standby]{1:25821:51519} [check] WARNING: PRIMARY not accessible
(attempt 1 of 3)
2020-11-16 15:31:55.270:[acfs_standby]{1:25821:51519} [check] WARNING: PRIMARY not accessible
(attempt 2 of 3)
2020-11-16 15:32:08.291:[acfs_standby]{1:25821:51519} [check] WARNING: PRIMARY not accessible
(attempt 3 of 3)
2020-11-16 15:32:08.291: [acfs_standby]{1:25821:51519} [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 `'/sbin/acfsutil repl util verifyprimary /mnt/acfs_repl -v'` to determine why the primary hub is inaccessible. Refer to [Appendix F](#) for common AFS Replication errors and their solutions.

- If the primary file system cluster is down and cannot be restarted, issue an ACFS failover on the standby hub:

```
$ /sbin/acfsutil repl failover /mnt/acfs_repl          # Specify the correct mount point
$ /sbin/acfsutil repl info -c -v /mnt/acfs_repl
```

- Run the following commands to prepare the `acfs_primary` resource to start on the new primary hub, and then restart GoldenGate:

```
$ echo "RESTART" > $MOUNT_POINT/acfs_primary
$ agctl start goldengate SOURCE                      # Specify the GoldenGate instance name
```

- 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 communicate with each other.

Use case #3 – Double Failure Case: Primary Hub Failure and Standby Connectivity Failure

If the primary hub 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_oracle.trc`) on the active cluster node:

```
2020-11-16 17:08:52.621:[acfs_primary]{1:40360:36312} [start] WARNING: PRIMARY file system
/mnt/acfs_repl previously crashed
2020-11-16 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:

```
2020-11-16 17:25:54.224:[acfs_primary]{1:40360:37687} [start] WARNING: PRIMARY /mnt/acfs_repl
disabled to prevent split brain
```

Use the following action plan with this scenario.

- Check the standby file system, using the command `'/sbin/acfsutil repl util verifystandby /mnt/acfs_repl -v'` to determine why the standby hub is inaccessible. Refer to [Appendix F](#) for common AFS Replication errors and their solutions.
- If communication with the the standby file system can re-established, restart GoldenGate on the primary hub:

```
$ agctl start goldengate SOURCE                      # Specify the GoldenGate instance name
```

- If communication with the standby file system cannot be re-established, use the following commands to restart GoldenGate on the primary hub:

```
$ echo "RESTART" > $MOUNT_POINT/acfs_primary
$ agctl start goldengate SOURCE                      # Specify the GoldenGate instance name
```

- When communication with the standby file system is restored, ACFS Replication will continue to replicate primary file system changes.

CONCLUSION

To summarize, GoldenGate provides a Microservices architecture that can be easily deployed in a hub configuration to minimize the number of components deployed on the database servers while providing the highest level of Maximum Availability Architecture (MAA).

Following the guidelines and best practices provided in this technical brief, the hub configuration can be easily deployed in the most robust and reliable HA configuration possible by utilizing Oracle Grid Infrastructure and ACFS to ensure both primary and standby hubs are protected themselves from disaster. It is assumed that Oracle RAC and Oracle Active Data Guard would already be configured for the source and target databases, providing the additional layers of protection, redundancy and disaster recovery.

APPENDIX A: EXAMPLE SSH DAEMON RESTART ACTION SCRIPT

Use the example CRS action script to restart the `ssh` daemon, making sure the script is owned by the Grid Infrastructure user.

```
#!/bin/bash

LOGGER="/bin/logger -t sshd_restart"
PATH_NAME=/tmp/sshd_restarted

LOGGER_FACILITY=user
PERL_ALARM_TIMEOUT=14
GREP=/bin/grep
ECHO=/bin/echo
BASENAME=/bin/basename
TOUCH=/bin/touch
RM="/bin/rm -rf"

### ensure messages are displayed in English for pattern matching
LANG=en_US.UTF-8
NLS_LANG=American_America.AL32UTF8

export PERL_ALARM_TIMEOUT
export STATUS_TIMEOUT
export LANG NLS_LANG

if [ -z "$STATUS_TIMEOUT" ]; then STATUS_TIMEOUT=0; fi

logit () {
    ### type: info, error, debug
    type=$1
    msg=$2
    if [ "$type" = "info" ]; then
        $ECHO $msg
        $LOGGER -p ${LOGGER_FACILITY}.info "$msg"
    elif [ "$type" = "error" ]; then
        $ECHO $msg
        $LOGGER -p ${LOGGER_FACILITY}.error "$msg"
    elif [ "$type" = "debug" ]; then
        $ECHO $msg
        $LOGGER -p ${LOGGER_FACILITY}.debug "$msg"
    fi
}

### determine how we were called, derive location
SCRIPTPATH=$0
SCRIPTNAME=`$BASENAME $SCRIPTPATH`

$ECHO $SCRIPTPATH | $GREP ^/ > /dev/null 2>&1

if [ $? -ne 0 ]; then
    MYDIR=`pwd`
    SCRIPTPATH=${MYDIR}/${SCRIPTPATH}
fi

case "$1" in
'start')
    logit info "Restarting sshd..."
    $RM $PATH_NAME
```

```

/bin/systemctl restart sshd
$ECHO "STARTED" > $PATH_NAME
exit 0
;;

'check'|'status')
  STARTED="$((${GREP} STARTED ${PATH_NAME})|wc -l 2>&1)";
  if [ $STARTED -ne 1 ]; then
    logit info "sshd restart is stopped"
    exit 1
  fi
  $SCRIPTPATH start
  logit info "sshd restarted"
  ;;

'restart')
  logit info "Restarting sshd restart"
  $SCRIPTPATH start
  ;;

'stop')
  logit info "Stopping sshd restart..."
  $ECHO "STOPPED" > $PATH_NAME
  exit 0
  ;;

'clean'|'abort')
  logit info "Stopping sshd restart..."
  $ECHO "STOPPED" > $PATH_NAME
  exit 0
  ;;

*)
  $ECHO "Usage: $SCRIPTNAME { start | stop | check | status | restart | clean | abort }"
  ;;
esac

```

APPENDIX B: ACFS_PRIMARY CRS ACTION SCRIPT

The following example CRS action script checks that the ACFS file system is mounted, that it is an ACFS primary file system, and that the standby file system verification is successful. If the standby file system fails its verification check, warning messages are printed to the CRS trace file (crsd_scriptagent_oracle.trc).

Copy the script to all of the primary and standby RAC nodes that are part of the ACFS and GoldenGate environment, owned by the Grid Infrastructure user. Set the file permissions to 755.

```
#!/bin/bash
# acfs action.scr - Checks if ACFS file system is of type PRIMARY (0 -
#                   success/online) or STANDBY (1 - failure/offline)
#                   GoldenGate managed by XAG, can only start if this returns 0.
# Replace with correct replicated ACFS mount point
MOUNT_POINT=/mnt/acfs_rep1
LOGGER="/bin/logger -t ACFS_GG01"
PATH_NAME=$MOUNT_POINT/acfs primary

# NOTE: When creating the CRS resource, set the CHECK_TIMEOUT and
#       CHECK_INTERVAL > (ATTEMPTS*INTERVAL)
ATTEMPTS=3      # Number of tries to check the remote standby file system
INTERVAL=10     # Number of seconds between each attempt

LOGGER_FACILITY=user
PERL_ALARM_TIMEOUT=14
GREP=/bin/grep
ECHO=/bin/echo
SLEEP=/bin/sleep
BASENAME=/bin/basename
STAT=/usr/bin/stat
PERL=/usr/bin/perl
TOUCH=/bin/touch
RM="/bin/rm -rf"

### ensure messages are displayed in English for pattern matching
LANG=en_US.UTF-8
NLS_LANG=American_America.AL32UTF8
export STAT MOUNT_POINT PERL_ALARM_TIMEOUT
export STATUS_TIMEOUT
export LANG NLS_LANG
if [ -z "$STATUS_TIMEOUT" ]; then STATUS_TIMEOUT=0; fi
logit () {
    ### type: info, error, debug
    type=$1
    msg=$2
    if [ "$type" = "info" ]; then
        $ECHO $msg
        $LOGGER -p ${LOGGER_FACILITY}.info "$msg"
    elif [ "$type" = "error" ]; then
        $ECHO $msg
        $LOGGER -p ${LOGGER_FACILITY}.error "$msg"
    elif [ "$type" = "debug" ]; then
        $ECHO $msg
        $LOGGER -p ${LOGGER_FACILITY}.debug "$msg"
```

```

    fi
}
### determine how we were called, derive location
SCRIPTPATH=$0
SCRIPTNAME=`$BASENAME $SCRIPTPATH`

$ECHO $SCRIPTPATH | $GREP ^/ > /dev/null 2>&1
if [ $? -ne 0 ]; then
    MYDIR=`pwd`
    SCRIPTPATH=${MYDIR}/${SCRIPTPATH}
fi

function check_for_standby()
{
$PERL <<'TOT'
    eval {
        $STANDBY=`/sbin/acfsutil repl info -c -v /mnt/acfs_repl|grep "Site"|grep Standby|wc -l 2>&1`;
        if ( $STANDBY == 1 ) {
            exit 1;
        } else {
            exit 0;
        }
    };
TOT
    RC=$?

    ### process return codes from the perl block
    if [ $RC -eq 1 ]; then
        echo 1
    else
        echo 0
    fi
}

# standby_status - Verify the standby DS is accessible and in good standing
function standby_status()
{
    OUTPUT="$(ls -l)"
    s_status="$(/sbin/acfsutil repl util verifystandby ${MOUNT_POINT}|cut -d ' ' -f4 2>&1)"

    if [ $s_status -ne 0 ]; then
        ### FS is not in OKAY status
        echo $s_status
    else # FS is OKAY
        echo 0;
    fi
}

# verify_standby - tries to determine current status of standby
function verify_standby()
{
    for ((i=1;i<=$ATTEMPTS;i++)); do

```

```

sresult=$(standby_status)                # Check result is 0 (SUCCESS)

if [ $sresult -eq 0 ]; then
    break
else
    logit debug "WARNING: STANDBY not accessible (attempt $i of $ATTEMPTS)"
    if [ $i -lt $ATTEMPTS ]; then
        $SLEEP $INTERVAL
    fi
fi
done

if [ $i -lt $ATTEMPTS ]; then
    logit debug "SUCCESS: Remote STANDBY file system $MOUNT_POINT is ONLINE"
    presult=0
else
    logit debug "WARNING: Problem with STANDBY file system (error: $sresult)"
    presult=0
fi
}

function verify_primary()
{
    ### check to see if ACFS mounted
    $PERL <<'TOT'
    $timeout = $ENV{'PERL_ALARM_TIMEOUT'};
    $SIG{ALRM} = sub {
        ### we have a problem and need to cleanup
        exit 3;
        die "timeout" ;
    };
    alarm $timeout;
    eval {
        $STATUSOUT=`$ENV{'STAT'} -f -c "%T" $ENV{'MOUNT_POINT'} 2>&1 `;

        chomp($STATUSOUT);
        ### added fuseblk check for Linux 6 output
        if ( ( $STATUSOUT eq 'acfs' ) ) {
            ### FS is mounted
            exit 0
        } else {
            ### filesystem is offline
            exit 1;
        }
    };
    TOT

    RC=$?
    presult=0

    ### process return codes from the perl block
    if [ $RC -eq 3 ]; then

```

```

STATUS_TIMEOUT=$(( $STATUS_TIMEOUT + 1 ))
logit error "Found timeout while checking status, cleaning mount automatically"
$SCRIPTPATH clean
logit debug "Filesystem is OFFLINE"
presult=1
elif [ $SRC -eq 2 ]; then
    STATUS_TIMEOUT=$(( $STATUS_TIMEOUT + 1 ))
    logit error "Found error while checking status, cleaning mount automatically"
    $SCRIPTPATH clean
    logit debug "WARNING: PRIMARY file system $MOUNT_POINT OFFLINE"
    presult=1
elif [ $SRC -eq 1 ]; then
    logit debug "WARNING: PRIMARY file System is not mounted"
    presult=2
elif [ $SRC -eq 0 ]; then      # File system mounted
    result=$(check_for_standby)      # Determine if FS primary or standby
    if [ $result -eq 1 ]; then
        presult=1
    fi
fi # end if file system mounted
}

case "$1" in
'start')
    logit info "$SCRIPTNAME starting at $MOUNT_POINT"

    result=$(check_for_standby)      # Return 0 if NOT standby file system

    if [ $result -eq 0 ]; then
        verify_primary
    else
        logit info "Detected local standby file system"
        exit 1
    fi

    if [[ $presult == 0 ]]; then
# Primary file system is mounted R/W
# Check to see if $PATH_NAME is "STARTED", indicating a previous CRASH of the node

        STARTED="$( ${GREP} STARTED ${PATH_NAME} | wc -l 2>&1)";

        if [ $STARTED -eq 1 ]; then      # File system was previously crashed
            logit info "WARNING: PRIMARY file system $MOUNT_POINT previously crashed"

# Check standby file system is accessible before starting
            sresult=$(standby_status)      # Check STANDBY result is 0 (SUCCESS)

            if [ $sresult -ne 0 ]; then
                logit debug "WARNING: STANDBY not accessible - disabling acfs_primary"
                $ECHO "DISABLED" > $PATH_NAME
                exit 0
            fi
        fi
    fi

```



```

        fi
    fi

    DISABLED="$( ${GREP} DISABLED ${PATH_NAME}|wc -l 2>&1)";
    if [ $DISABLED -eq 1 ]; then
# Check to see if standby is accessible yet
        sresult=$(standby_status)           # Check STANDBY result is 0 (SUCCESS)

        if [ $sresult -ne 0 ]; then
# Standby still not accessible
            logit info "WARNING: PRIMARY $MOUNT_POINT disabled to prevent split brain"
            exit 0
        fi          # else will will allow primary to restart because standby is not back
    fi

    $RM $PATH_NAME
    $ECHO "STARTED" > $PATH_NAME

# Check to see if primary status file exists
    if [ -f "$PATH_NAME" ]
    then          # Will only get called for resource START
        logit info "acfs_primary started"

# IF a GoldenGate Distribution Server sending trails to Receiver Server on the Hub, change the path
# target:
#     /u01/oracle/scripts/change_path_hub.sh scam09db05:443 scam09db06:443 scam_to_gghub gg01
# /u01/oracle/scripts/scam.cfg
#     RET=$?
#     if [[ $RET == 0 ]]; then
#         logit info "Changed distribution path scam_to_gghub target URI"
#     else
#         logit info "ERROR: Unable to change distribution path target URI"
#     fi
#     exit 0
# else
#     logit info "WARNING: Unable to create $PATH_NAME"
#     exit 1
# fi
# else
#     logit info "WARNING: Verify of PRIMARY failed"
#     exit 1
# fi
;;

'check'|'status')
    result=$(check_for_standby)    # Return 0 if NOT standby file system

    if [ $result -ne 0 ]; then
        logit info "Detected local standby file system"
        exit 1
    fi

```

```

verify_primary      # Check status of primary FS

logit info "DEBUG: presult $presult"

if [[ $presult == 0 ]]; then          # Primary file system is good

# First check that it wasn't manually STOPPED:
  STOPPED="$((${GREP} STOPPED ${PATH_NAME}|wc -l 2>&1)";
  if [ $STOPPED -eq 1 ]; then
    logit info "SUCCESS: PRIMARY file system $MOUNT_POINT is STOPPED"
    exit 1
  fi

  DISABLED="$((${GREP} DISABLED ${PATH_NAME}|wc -l 2>&1)";
  if [ $DISABLED -eq 1 ]; then
    logit info "WARNING: Primary file system $MOUNT_POINT DISABLED"
    exit 1
  fi

  RESTART="$((${GREP} RESTART ${PATH_NAME}|wc -l 2>&1)";
  if [ $RESTART -eq 1 ]; then
    logit info "Restarting PRIMARY file system $MOUNT_POINT"
    $SCRIPTPATH start
  fi

sresult=$(standby_status)            # Check standby status (0=GOOD)

if [ $sresult -eq 0 ]; then          # Standby all good
  if [ -f "$PATH_NAME" ]
  then
    STARTED="$((${GREP} STARTED ${PATH_NAME}|wc -l 2>&1)";
    if [ $STARTED -eq 1 ]; then
      logit info "SUCCESS: STANDBY file system $MOUNT_POINT is ONLINE"
      exit 0
    fi
  else # Can't access $PATH_NAME
    logit info "WARNING: PRIMARY file $PATH_NAME does NOT exist "
    exit 1
  fi
  else # $sresult != 0 (BAD)
    verify_standby # Get current status of standby file system and report findings in CRS trace
  fi
elif [ $presult -eq 2 ]; then        # Primary file system not mounted, maybe standby FS
  exit 1
else
  logit info "WARNING: Problem with local PRIMARY file system."
  exit 1
fi
;;

'restart')
  logit info "Restart -- ACFS file system type checking..."

```

```

$SCRIPTPATH start
;;

'stop')
  logit info "Stop -- Stopping ACFS file system type checking..."
  $ECHO "STOPPED" > $PATH_NAME
  exit 0
  ;;

'clean'|'abort')
  logit info "Clean/Abort -- Stopping ACFS file system type checking..."
  $ECHO "STOPPED" > $PATH_NAME
  exit 0
  ;;

*)
  $ECHO "Usage: $SCRIPTNAME { start | stop | check | status | restart | clean | abort }"
  ;;

esac

```

APPENDIX C: ACFS_STANDBY CRS ACTION SCRIPT

The following example `acfs_standby` action script checks that the ACFS file system is mounted, that it is an ACFS standby file system, and that the primary file system verification is successful.

Copy the script to all of the primary and standby RAC nodes that are part of the ACFS and GoldenGate environment, owned by the Grid Infrastructure user. Set the file permissions to 755.

```
#!/bin/bash

# Modify the parameters in this section to match your environment for ACFS
# replication
MOUNT_POINT=/mnt/acfs_rep1
LOGGER="/bin/logger -t ACFS GG01"
PATH_NAME=/tmp/acfs_standby

# NOTE: When creating the CRS resource, set the CHECK TIMEOUT and CHECK INTERVAL >
#       (ATTEMPTS*INTERVAL)
ATTEMPTS=3      # Number of tries to check the remote primary file system
INTERVAL=10     # Number of seconds between each attempt

LOGGER_FACILITY=user
PERL_ALARM_TIMEOUT=14
ECHO=/bin/echo
GREP=/bin/grep
SLEEP=/bin/sleep
BASENAME=/bin/basename
STAT=/usr/bin/stat
PERL=/usr/bin/perl
TOUCH=/bin/touch
RM="/bin/rm -rf"

### ensure messages are displayed in English for pattern matching
LANG=en_US.UTF-8
NLS_LANG=American_America.AL32UTF8

export STAT MOUNT_POINT PERL_ALARM_TIMEOUT
export STATUS_TIMEOUT ATTEMPTS INTERVAL
export LANG NLS_LANG

if [ -z "$STATUS_TIMEOUT" ]; then STATUS_TIMEOUT=0; fi

logit () {
    ### type: info, error, debug
    type=$1
    msg=$2
    if [ "$type" = "info" ]; then
        $ECHO $msg
        $LOGGER -p ${LOGGER_FACILITY}.info "$msg"
    elif [ "$type" = "error" ]; then
        $ECHO $msg
        $LOGGER -p ${LOGGER_FACILITY}.error "$msg"
    elif [ "$type" = "debug" ]; then
        $ECHO $msg
    fi
}
```

```

    $LOGGER -p ${LOGGER_FACILITY}.debug "$msg"
fi
}

### determine how we were called, derive location
SCRIPTPATH=$0
SCRIPTNAME=`$BASENAME $SCRIPTPATH`

$ECHO $SCRIPTPATH | $GREP ^/ > /dev/null 2>&1
if [ $? -ne 0 ]; then
    MYDIR=`pwd`
    SCRIPTPATH=${MYDIR}/${SCRIPTPATH}
fi

# check_for_standby - checks that the ACFS mount point is a STANDBY (return 1)
function check_for_standby()
{
$PERL <<'TOT'
    eval {
        $STANDBY=`/sbin/acfsutil repl info -c -v $ENV{'MOUNT_POINT'}|grep "Site"|grep Standby|wc -l
2>&1`;

        if ( $STANDBY == 1 ) {
            ### FS is a STANDBY
            exit 1;
        } else { # FS is a PRIMARY
            exit 0;
        }
    };
TOT
    RC=$?

    ### process return codes from the perl block
    if [ $RC -eq 1 ]; then
        echo 1
    else
        echo 0
    fi
}

# verify_primary - check the primary FS status
function verify_primary()
{
$PERL <<'TOT'
    eval {
        $p_status=`/sbin/acfsutil repl util verifyprimary $ENV{'MOUNT_POINT'}|cut -d ' ' -f4 2>&1`;

        if ( $p_status > 0 ) { # Primary NOT okay!
            ### FS is not in OKAY status
            exit $p_status;
        } else { # FS is OKAY
            exit 0;

```

```

    }
};

TOT
RC=$?

### process return codes from the perl block
if [ $RC -ne 1 ]; then
    echo $RC
else
    echo 0
fi
}

case "$1" in
'start')
    logit info "$SCRIPTNAME starting to check ACFS remote primary at $MOUNT_POINT"

    result=$(check_for_standby)

    if [ $result -ne 1 ]; then # STANDBY file system detected
        logit info "local file system ($MOUNT_POINT) is a PRIMARY"
    fi

    $RM $PATH_NAME
    $ECHO "STARTED" > $PATH_NAME
    if [ -f "$PATH_NAME" ]
    then
        # Will only get called for resource start, and NOT changing from offline to online
        exit 0
    else
        logit info "ERROR: Unable to create standby file system status file ($PATH_NAME)"
        exit 1
    fi
;;

'check'|'status')
    ### Check to see if ACFS mounted as STANDBY file system
    $PERL <<'TOT'
        $timeout = $ENV{'PERL_ALARM_TIMEOUT'};
        $SIG{ALRM} = sub {
            ### we have a problem and need to cleanup
            exit 3;
            die "timeout" ;
        };
        alarm $timeout;
        eval {
            $STATUSOUT=`$ENV{'STAT'} -f -c "%T" $ENV{'MOUNT_POINT'} 2>&1`;
            chomp($STATUSOUT);

            if ( ( $STATUSOUT eq 'acfs' ) ) {
                ### filesystem is mounted
                exit 0
            } else {

```

```

        ### filesystem is offline
        exit 1;
    }
};
TOT

RC=$?
### process return codes from the perl block
if [ $RC -eq 3 ]; then
    STATUS_TIMEOUT=$(( $STATUS_TIMEOUT + 1 ))
    logit error "Found timeout while checking status, cleaning mount automatically"
    $SCRIPTPATH clean
    logit debug "File system check encountered a problem"
    exit 1
elif [ $RC -eq 1 ]; then
    logit debug "File system is OFFLINE"
    exit 1
elif [ $RC -eq 0 ]; then
    # File system is MOUNTED

    result=$(check_for_standby)          # 0:Primary 1:Standby

    if [ $result -eq 1 ]; then # STANDBY file system detected

# Check that primary FS is accessible. This is optional and can be skipped if
# want to continue operating if primary FS is in trouble - SEE WARNING ABOVE!

        for ((i=1;i<=$ATTEMPTS;i++)); do
            presult=$(verify_primary)          # Check result is 0 (SUCCESS)

            if [ $presult == 0 ]; then # Primary file system OKAY
                logit debug "SUCCESS: PRIMARY file system $MOUNT_POINT is ONLINE"
                exit 0
            elif [ $presult -eq 222 ] || [ $presult -eq 255 ] || [ $presult -eq 237 ]; then
                logit debug "WARNING: PRIMARY not accessible (attempt $i of $ATTEMPTS)"
                if [ $i -lt $ATTEMPTS ]; then
                    $SLEEP $INTERVAL
                fi
            fi
        done
        logit debug "WARNING: Problem with PRIMARY file system (error: $presult)"
        exit 0
    else
        logit debug "Local PRIMARY file system $MOUNT_POINT"
        exit 1
    fi # if [ $result -eq 1 ]; then # STANDBY file system detected
fi # if [ $RC -eq 3 ];
;;

'restart')
    logit info "Restarting ACFS remote primary checking..."
    $SCRIPTPATH start
;;

```



```
'stop')
  logit info "Stopping ACFS remote primary checking..."
  $ECHO "STOPPED" > $PATH_NAME
  exit 0
;;

'clean'|'abort')
  logit info "Stopping ACFS file system type checking..."
  $ECHO "STOPPED" > $PATH_NAME
  exit 0
;;

*)
  $ECHO "Usage: $SCRIPTNAME { start | stop | check | status | restart | clean | abort }"
  ;;
esac
```

APPENDIX D: DISTRIBUTION PATH TARGET CHANGE SCRIPT

After a hub role transition, 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. This example assumes the source GoldenGate deployment is configured in a MAA architecture, and so the current location of the distribution server must be determined.

```
#!/bin/bash

# change_path_gghub.sh - changes the target host of a GG Distribution Path when
# the target moves to GG Hub standby.

# Example usage: ./change_path_gghub.sh scam09db05:443 scam09db06:443 scam to gghub gg01
/u01/oracle/scripts/gghub/scam.cfg

SOURCE1=$1      # scam09db05:443
SOURCE2=$2      # scam09db06:443
DPATH=$3        # scam_to_gghub
DEP=$4          # gg01
ACCESS=$5       # scam.cfg

CONNECT=0

result=$(curl -i -s -K $ACCESS -k https://$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 "*** Couldn't contact Distribution Server at $SOURCE1 ***"

# Try the other source host:
else
    echo "*** Got status of Distribution Server at $SOURCE1 ***"
    SOURCE=$SOURCE1
    CONNECT=1
fi

if [ $CONNECT -eq 0 ]; then

    result=$(curl -i -s -K $ACCESS -k https://$SOURCE2/$DEP/distsrvr/services/v2/sources/$DPATH -X
GET | grep HTTP | awk '{print $2}')

    if [[ -z $result || $result -ne 200 ]]; then # Managed to access the Distr Server
        echo "*** Couldn't contact Distribution Server at $SOURCE2 ***"
# Try the other source host:
    else
        echo "*** Got status of Distribution Server at $SOURCE2 ***"
        SOURCE=$SOURCE2
        CONNECT=1
    fi
fi

if [ $CONNECT -eq 1 ]; then
    curl -s -K $ACCESS -k https://$SOURCE/$DEP/distsrvr/services/v2/sources/$DPATH -X PATCH --data
'{"status": "stopped"}' | python -m json.tool
```

```
    curl -s -K $ACCESS -k https://$SOURCE/$DEP/distsrvr/services/v2/sources/$DPATH -X PATCH --data
'{"target":{"uri":"ogg://scao04client07-vip1:9103/services/v2/targets?trail=dd"}}' | python -m
json.tool

    curl -s -K $ACCESS -k https://$SOURCE/$DEP/distsrvr/services/v2/sources/$DPATH -X GET | python -m
json.tool

    curl -s -K $ACCESS -k https://$SOURCE/$DEP/distsrvr/services/v2/sources/$DPATH -X PATCH --data
'{"status": "running"}'| python -m json.tool

    exit 0
fi

# If here, means we couldn't connect to either Distribution Servers
exit 1
```

APPENDIX E: EXAMPLE GOLDENGATE DEPLOYMENT CREATION RESPONSE FILES

Most of the configuration variables are related to file system directories where the deployment and GoldenGate process files will be stored, so it is important that the directories match those already created.

Use the correct response file from the current GoldenGate installation software to ensure the parameters are specified correctly.

Example response file parameter changes required:

Section A - General

DEPLOYMENT_NAME=TARGET # Make sure this is a unique deployment name

Section B - Administrator Account

ADMINISTRATOR_USER=admin # GG deployment administration account, NOT a database account.

ADMINISTRATOR_PASSWORD=<password> # Note: you MUST specify a password

Section C - Service Manager

SERVICEMANAGER_DEPLOYMENT_HOME=/mnt/acfs_rep1/deployments/ggsm # File system directory for the Service Manager deployment. This only gets set for the first deployment creation. Leave blank for subsequent deployment creations.

HOST_SERVICEMANAGER=127.0.0.1 # Leave as localhost (127.0.0.1)

PORT_SERVICEMANAGER=9100 # Set to an unused port number, unique for this deployment. For the second deployment creation, set this to the port# used for the initial deployment creation.

SECURITY_ENABLED=false # Security is handled by NGINX reverse proxy.

CREATE_NEW_SERVICEMANAGER=true # Only set to a true for first deployment creation. For subsequent deployment creations, set to false.

INTEGRATE_SERVICEMANAGER_WITH_XAG=true

Section D - Software Home

OGG_SOFTWARE_HOME=/u01/app/goldengate/db19c # GoldenGate software location

Section E - Deployment Directories

OGG_DEPLOYMENT_HOME=/mnt/acfs_rep1/deployments/target # MUST be stored in the ACFS replicated file system

Section F - Environment Variables

ENV_ORACLE_HOME=/u01/app/oracle/product/19.1/client

ENV_LD_LIBRARY_PATH=\${ORACLE_HOME}/lib:/u01/app/oracle/product/19.1/client/lib:/u01/app/oracle/product/19.1/client/rdbms/lib:/u01/app/oracle/product/19.1/client/jdbc/lib:/u01/oracle/goldengate/target/lib

ENV_TNS_ADMIN=/u01/app/oracle/network/admin # Use a single TNS_ADMIN for all client installations to make admin of sqlnet.ora and tnsnames.ora easier

Section I - Services

PORT_ADMINSRVR=9101 # Set to an unused port number, unique for this deployment.

DISTRIBUTION_SERVER_ENABLED=false # Not using due to local trail files.

PORT_DISTSRVR=9102 # Set to an unused port number, unique for this deployment.

RECEIVER_SERVER_ENABLED=false # Not using due to local trail files.

PORT_RCVRSRVR=9103 # Set to an unused port number, unique for this deployment.

PORT_PMSRVR=9104 # Set to an unused port number, unique for this deployment.

UDP_PORT_PMSRVR=9105 # Set to an unused port number, unique for this deployment.

PMSRVR_DATASTORE_TYPE=BDB # Or use LMDB

PMSRVR_DATASTORE_HOME= # If using BDB, set the directory outside of ACFS mount point.

Section J - REPLICATION OPTIONS

OGG_SCHEMA=ggadmin # Set to schema used for GG administrator in the database. MAKE SURE SOURCE & TARGET DATABASE SCHEMA NAMES MATCH.

For subsequent deployment creations when using the same OGG_HOME directory, the following response file parameters will be different:

Section A - General

DEPLOYMENT_NAME=SOURCE # Make sure this is a unique deployment name, should use SOURCE or TARGET

Section C - Service Manager

SERVICEMANAGER_DEPLOYMENT_HOME= # Leave blank for subsequent deployment creations.

HOST_SERVICEMANAGER=127.0.0.1 # MUST leave as localhost (127.0.0.1)

PORT_SERVICEMANAGER=9100 # Set this to the port# used for the initial deployment creation.

CREATE_NEW_SERVICEMANAGER=false # Initial deployment already created the Service Manager.

EXISTING_SERVICEMANAGER_IS_XAG_ENABLED=true

Section D - Software Home

OGG_SOFTWARE_HOME=/u01/app/oracle/goldengate/db12c # GoldenGate software location specified above in step #2 for the correct database compatibility,

Section E - Deployment Directories

OGG_DEPLOYMENT_HOME=/mnt/acfs_rep1/deployments/source # Store on the replicated ACFS file system.

Section F - Environment Variables

ENV_ORACLE_HOME=/u01/app/oracle/product/12.2/client # Assuming the source database is 12.2, but this may differ depending on the database version

ENV_LD_LIBRARY_PATH=\${ORACLE_HOME}/lib:/u01/app/oracle/product/19.1/client/lib:/u01/app/oracle/product/19.1/client/rdbms/lib:/u01/app/oracle/product/19.1/client/jdbc/lib:/u01/app/oracle/goldengate/source/lib

ENV_TNS_ADMIN=/u01/app/oracle/network/admin # Use a single TNS_ADMIN for all client installations to make admin of sqlnet.ora and tnsnames.ora easier

Section I - Services

PORT_ADMINSRVR=9111 # Set to an unused port number, unique for this deployment

DISTRIBUTION_SERVER_ENABLED=false

PORT_DISTSRVR=9112 # Set to an unused port number, unique for this deployment

RECEIVER_SERVER_ENABLED=false

PORT_RCVRSRVR=9113 # Set to an unused port number, unique for this deployment

PORT_PMSRVR=9114 # Set to an unused port number, unique for this deployment

UDP_PORT_PMSRVR=9115 # Set to an unused port number, unique for this deployment

PMSRVR_DATASTORE_TYPE=BDB # Or use LMDB

PMSRVR_DATASTORE_HOME=

If using BDB, set the directory outside of ACFS mount point.

APPENDIX F: TROUBLESHOOTING

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 provided in Appendix B and Appendix C, 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.

For example:

```
$ acfsutil repl util verifyprimary /mnt/acfs_rep1 -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 hosts. Once the problem has been resolved, rerun the verify commands to ensure there are no further problems.

If you are using an automated file system failover capability like the example in Appendix C, 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/<node name>/crs/trace`.

Below are some common failures that can occur with incorrect ACFS replication configuration.

1. SSH daemon is shutdown or not configured to run on the VIP

When using an application VIP on the ACFS primary and standby clusters, 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: `sbin/acfsutil repl util verifystandby /mnt/acfs_rep1 -v`

```
- Attempting to ping scaqaa08client01-avip
- ping successful
- Attempting to ssh
  '/usr/bin/ssh -o BatchMode=true -o Ciphers=aes128-ctr -o ConnectTimeout=3 -x
oracle@scaqaa08client01-avip true 2>&1'
- ssh output: ssh: connect to host scaqaa08client01-avip port 22: Connection refused
- ssh output: ssh: connect to host scaqaa08client01-avip port 22: Connection refused
- ssh attempt failed, ret=255
verifystandby return code: 255
```

Every time the `verifyprimary` and `verifystandby` commands are run, the output is also written to a trace file in the CRS trace file directory. For example:


```

2020-06-11 17:08:58.905 : REPLCMP:3313712320: [246432]:0000:ofsrUtil: entered, isAdmin=1,
numargs=4, ac=5
2020-06-11 17:08:58.905 : REPLCMP:3313712320: [246432]:0001:ofsrGetSnapMount: Checking mount path
/mnt/acfs_repl
2020-06-11 17:08:58.906 : REPLCMP:3313712320: [246432]:0002:uscgGetReplicationState: repl state = 3
2020-06-11 17:08:58.906 : REPLCMP:3313712320: [246432]:0003:uscgGetRacoonV2: Interesting racoon
flags. USC_RACFLAGS=0x5
2020-06-11 17:08:58.906 : REPLCMP:3313712320: [246432]:0004:uscgMakeSshPath:
USC_SSHPATH=/usr/bin/ssh
2020-06-11 17:08:58.906 : REPLCMP:3313712320: [246432]:0005:uscgMakeAcfsutilPaths: Using ssh: TRUE
2020-06-11 17:08:58.906 : REPLCMP:3313712320: [246432]:0006:uscgInvokeCommand: invoking '/bin/ping
-c 1 -w 3 scaqaa08client01-avip > /dev/null 2>&1'
2020-06-11 17:08:58.911 : REPLCMP:3313712320: [246432]:0007:uscgVerifyStandbyV2: ping OK (/bin/ping
-c 1 -w 3 scaqaa08client01-avip > /dev/null 2>&1)
2020-06-11 17:08:58.921 : REPLCMP:3313712320: [246432]:0008:uscgVerifyStandbyV2: ssh produced
cmdOutput: ssh: connect to host scaqaa08client01-avip port 22: Connection refused
2020-06-11 17:09:00.932 : REPLCMP:3313712320: [246432]:0010:ERROR: uscgVerifyStandbyV2: could not
ssh to the standby (/usr/bin/ssh -o BatchMode=true -o Ciphers=aes128-ctr -o ConnectTimeout=3 -x
oracle@scaqaa08client01-avip true 2>&1), ret=255

```

When the 'connection refused' error is encountered, first check the VIP address is added to the /etc/ssh/sshd_config file:

```

# Determine the VIP address
$ crsctl stat res <VIP name> -f | grep USR_ORA_VIP |grep -v GEN

# Check the sshd_config file that the ListenAddress is correct:
ListenAddress <VIP network address>

```

Restart the ssh daemon (as root):

```

# /bin/systemctl restart sshd
# /bin/systemctl status sshd

```

Check that acfsutil repl verifystandby/verypriamary returns a result of '0' from both the primary and standby host.

2. Primary ACFS background resources are not running

When the primary ACFS background resources are not running, the verifypriamary command will fail with the following:

```

$ acfsutil repl util verifypriamary /mnt/acfs_repl -v
- Attempting to ping scao04client07-vip1
- ping successful
- Attempting to ssh
  '/usr/bin/ssh -o BatchMode=true -o Ciphers=aes128-ctr -o ConnectTimeout=3 -x
oracle@scao04client07-vip1 true 2>&1'
- ssh to primary was successful
- Checking primary file system via
  '/usr/bin/ssh -o BatchMode=true -o Ciphers=aes128-ctr -o ConnectTimeout=3 -x
oracle@scao04client07-vip1 /sbin/acfsutil info fs '/mnt/acfs_repl' > /dev/null 2>&1'
- Primary file system is mounted
- Checking primary file system availability
  '/usr/bin/ssh -o BatchMode=true -o Ciphers=aes128-ctr -o ConnectTimeout=3 -x
oracle@scao04client07-vip1 /sbin/acfsutil info fs '/mnt/acfs_repl'| grep state | grep Available >
/dev/null 2>&1'

```

```
- Primary file system is available
- Checking if the daemon is running via
  '/usr/bin/ssh -o BatchMode=true -o Ciphers=aes128-ctr -o ConnectTimeout=3 -x
oracle@scao04client07-vip1 /sbin/acfsutil repl bg info '/mnt/acfs_repl' | grep Current | grep
ONLINE > /dev/null 2>&1'

- Could not detect a running daemon
verifyprimary return code: 238
```

This error will also prevent a file system failover from occurring, reporting the following into an acfsutil trace file on the standby host:

```
2020-05-28 09:34:01.062 :REPLFAIL:3331837120: [62774] Start: /sbin/acfsutil.bin repl failover
/mnt/acfs_repl
2020-05-28 09:34:01.062 :REPLFAIL:3331837120: [62774]:0000:ofsrFailover: entered, isAdmin=0,
numargs=3, ac=3
2020-05-28 09:34:01.062 :REPLFAIL:3331837120: [62774]:0001:ofsrGetSnapMount: Checking mount path
/mnt/acfs_repl
...
2020-05-28 09:34:01.620 :REPLFAIL:3331837120: [62774]:0016:uscgVerifyPrimaryV2: file system is
available
2020-05-28 09:34:01.620 :REPLFAIL:3331837120: [62774]:0017:uscgInvokeCommand: invoking
'/usr/bin/ssh -o BatchMode=true -o Ciphers=aes128-ctr -o ConnectTimeout=3 -x
oracle@scao04client07-vip1 /sbin/acfsutil repl bg info '/mnt/acfs_repl' | grep Current | grep
ONLINE > /dev/null 2>&1'
2020-05-28 09:34:01.910 :REPLFAIL:3331837120: [62774]:0018:ERROR: uscgVerifyPrimaryV2: /usr/bin/ssh
-o BatchMode=true -o Ciphers=aes128-ctr -o ConnectTimeout=3 -x oracle@scao04client07-vip1
/sbin/acfsutil repl bg info '/mnt/acfs_repl' | grep Current | grep ONLINE > /dev/null 2>&1 did not
detect a running daemon
2020-05-28 09:34:01.910 :REPLFAIL:3331837120: [62774]:0019:ofsrFailover: using reverse, isAdmin=0
2020-05-28 09:34:01.910 :REPLFAIL:3331837120: [62774]:0020:uscgInvokeCommand: invoking
'/usr/bin/ssh -o BatchMode=true -o Ciphers=aes128-ctr -x oracle@scao04client07-vip1
/sbin/acfsutil repl reverse primary '/mnt/acfs_repl' > /dev/null'
2020-05-28 09:34:07.087 :REPLFAIL:3331837120: [62774] Returned: 1
```

Check the status of the ACFS background processes on the primary host:

```
$ /sbin/acfsutil repl bg info /mnt/acfs_repl

Resource:      ora.repl.dupd.datacl.acfs_repl.acfs
Target State:  ONLINE                      , OFFLINE
Current State: OFFLINE on pnode1, OFFLINE on pnode2
```

Restart the background services:

```
$ /sbin/acfsutil repl bg start /mnt/acfs_repl

$ /sbin/acfsutil repl bg info /mnt/acfs_repl
Resource:      ora.repl.dupd.datacl.acfs_repl.acfs
Target State:  OFFLINE                      , ONLINE
Current State: OFFLINE on pnode1, ONLINE on pnode2

$ /sbin/acfsutil repl info -c -v /mnt/acfs_repl

Site:                      Primary
Primary hostname:          pnode-vip1
Primary path:              /mnt/acfs_repl
```

```
Primary status: Running
Background Resources: Active
```

From the standby server, check primary file system status:

```
$ acfsutil repl util verifyprimary /mnt/acfs_repl
verifyprimary return code: 0
```

3. The primary or standby ACFS servers are not accessible

If the GoldenGate deployments have stopped running on the primary cluster for no apparent reason, first check the `crsd_scriptagent_oracle.trc` trace file located in the CRS trace directory. The deployments can be stopped due to the standby file cluster not being accessible from the primary. If this is the case, the following message will output into the trace file on the primary server:

```
2020-07-19 23:56:00.000 :CLSDYNAM:358348544: [acfs_primary]{1:17701:16727} [check] Standby not
accessible (attempt 1 of 3))
2020-07-19 23:56:13.233 :CLSDYNAM:358348544: [acfs_primary]{1:17701:16727} [check] Standby not
accessible (attempt 2 of 3))
2020-07-19 23:56:26.463 :CLSDYNAM:358348544: [acfs_primary]{1:17701:16727} [check] Standby not
accessible (attempt 3 of 3))
2020-07-19 23:56:26.463 :CLSDYNAM:358348544: [acfs_primary]{1:17701:16727} [check] Disabling
primary FS access via acfs_primary service
2020-07-19 23:56:26.463 :CLSDYNAM:358348544: [acfs_primary]{1:17701:16727} [check] ACFS primary
file system problem - manual intervention required
2020-07-19 23:56:26.464 : AGFW:362551040: [ INFO] {1:17701:16727} acfs_primary 1 1 state
changed from: ONLINE to: OFFLINE
```

From current primary server, verify the standby file system:

```
$ acfsutil repl util verifystandby /mnt/acfs_repl
verifystandby return code: 1
```

The output of the `acfsutil repl util verifystandby` command is shown on the screen, but also to an `acfsutil` trace file located in the CRS trace directory:

```
2020-07-19 23:56:10.110 : REPLCMP:3508260032: [333747]:0006:uscgInvokeCommand: invoking '/bin/ping
-c 1 -w 3 scao04client07-vip1 > /dev/null 2>&1'
2020-07-19 23:56:13.117 : REPLCMP:3508260032: [333747]:0007:ERROR: uscgVerifyStandbyV2: could not
ping (/bin/ping -c 1 -w 3 snode-vip > /dev/null 2>&1), ret=1
```

It is clear that something is stopping connectivity to the standby server. If the ping command is run manually, it too will fail:

```
$ /bin/ping -c 1 -w 3 snode-vip
PING snode-vip.example.com (1.234.56.789) 56(84) bytes of data.
--- snode-vip.example.com ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 2999ms
```

Make sure the standby server is running, and the VIP is running on both the primary and standby servers. Use the following command to check the status and start the VIP.

```
$ crsctl stat res gg_vip_prim
```

```
$ crsctl start res gg_vip_prmy
```

The acfs_primary resource should automatically restart on the hub primary cluster, but you can also be manually restarted, which in turn will restart the GoldenGate deployments:

```
$ crsctl start res acfs_primary
```

4. ACFS Replication ssh user problem

ACFS replication uses ssh to communicate between the primary and standby file systems. If there is a problem with the ACFS replication user on either the primary or standby server the acfsutil repl util verifyprimary/verifystandby commands will fail with the following error. This error is also reported in the crsd_scriptagent_oracle.trc trace file.

```
$ acfsutil repl util verifyprimary /mnt/acfs_repl -v
- Attempting to ping pnode-vip
- ping successful
- Attempting to ssh
  '/usr/bin/ssh -o BatchMode=true -o Ciphers=aes128-ctr -o ConnectTimeout=3 -x oracle@pnode-vip true 2>&1'
- ssh output: WARNING: Your password has expired.
- ssh output: Password change required but no TTY available.
- ssh output: WARNING: Your password has expired.
- ssh output: Password change required but no TTY available.
- ssh attempt failed, ret=1
verifyprimary return code: 255
```

The user problem can be confirmed trying to ssh to the primary host:

```
$ ssh oracle@pnode-vip
You are required to change your password immediately (password aged)
Last login: Thu Jun 11 20:24:13 PDT 2020
Last login: Thu Jun 11 20:26:06 2020 from pnode1.example.com
WARNING: Your password has expired.
You must change your password now and login again!
Changing password for user oracle.
Changing password for oracle.
```

Reset the user password on the primary host, and then retry the verify command:

```
$ acfsutil repl util verifyprimary /mnt/acfs_repl -v
- Attempting to ping pnode-vip
- ping successful
- Attempting to ssh
  '/usr/bin/ssh -o BatchMode=true -o Ciphers=aes128-ctr -o ConnectTimeout=3 -x oracle@pnode-vip true 2>&1'
- ssh to primary was successful
- Checking primary file system via
  '/usr/bin/ssh -o BatchMode=true -o Ciphers=aes128-ctr -o ConnectTimeout=3 -x oracle@pnode-vip /sbin/acfsutil info fs '/mnt/acfs repl' > /dev/null 2>&1'
- Primary file system is mounted
- Checking primary file system availability
  '/usr/bin/ssh -o BatchMode=true -o Ciphers=aes128-ctr -o ConnectTimeout=3 -x oracle@pnode-vip
```

```
/sbin/acfsutil info fs '/mnt/acfs_repl' | grep state | grep Available > /dev/null 2>&1'
- Primary file system is available
- Checking if the daemon is running via
  '/usr/bin/ssh -o BatchMode=true -o Ciphers=aes128-ctr -o ConnectTimeout=3 -x oracle@pnode-vip
/sbin/acfsutil repl bg info '/mnt/acfs_repl' | grep Current | grep ONLINE > /dev/null 2>&1'
- Daemon is ONLINE
verifyprimary return code: 0
```

5. SSH Host key verification problem

When a problem is encountered with the ssh key changing on a primary or standby file system server ACFS Replication will fail with the following error.

```
$ acfsutil repl util verifyprimary /mnt/acfs_repl -v
- Attempting to ping pnode-vip
- ping successful
- Attempting to ssh
  '/usr/bin/ssh -o BatchMode=true -o Ciphers=aes128-ctr -o ConnectTimeout=3 -x oracle@pnode-vip
true 2>&1'
- ssh output: Host key verification failed.
- ssh attempt failed, ret=255
verifyprimary return code: 255
```

The error can be resolved by manually issuing the `ssh` command to the current primary or standby server that reported the failure, as shown below.

```
$ ssh oracle@pnode-vip

The authenticity of host 'pnode-vip (1.234.567.89)' can't be established.
ECDSA key fingerprint is SHA256:itKt1FeSjPESSfi4LVboJAljXAHwVdQHh+8racY/SNE.
ECDSA key fingerprint is MD5:c0:29:ea:db:f0:d2:7f:81:07:a1:7a:df:22:e3:24:b6.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'pnode-vip,1..234.567.89' (ECDSA) to the list of known hosts.
Last login: Tue Jun 30 11:44:59 PDT 2020 from client01.example.com on pts/2
Last login: Tue Jun 30 11:54:04 2020 from pnode1.example.com
```

Confirm ssh now works correctly:

```
$ acfsutil repl util verifyprimary /mnt/acfs_repl
verifyprimary return code: 0
```

Troubleshooting 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.

1. XAG log file

Location: <XAG installation directory>/log/<hostname>

Example location: /u01/oracle/xag/log/pnode1

File name: agctl_goldengate_oracle.trc

Contains all commands executed with agctl along with the output from the commands, including those commands that are executed by CRS.

Example:

```
2020-07-16 20:26:48: agctl start goldengate TARGET
2020-07-16 20:26:48: executing cmd: /u01/app/19.0.0.0/grid/bin/crsctl status res
xag.TARGET.goldengate
2020-07-16 20:26:48: executing cmd: /u01/app/19.0.0.0/grid/bin/crsctl status res
xag.TARGET.goldengate -f
2020-07-16 20:26:49: executing cmd: /u01/app/19.0.0.0/grid/bin/crsctl start resource
xag.TARGET.goldengate -f
2020-07-16 20:26:55: Command output:
> CRS-2672: Attempting to start 'xag.TARGET.goldengate' on 'pnode1'
> CRS-2676: Start of 'xag.TARGET.goldengate' on 'pnode1' succeeded
>End Command output
```

2. XAG GoldenGate instance trace file

Location: <XAG installation directory>/log/<hostname>

Example location: /u01/oracle/xag/log/pnode1

File name: <GoldenGate_instance_name>_agent_goldengate.trc

Contains the output from the commands executed by agctl along with the environment variables used, and any debug output enabled for the underlying commands.

Example excerpt:

```
2020-07-19 23:02:37: Getting attributes for xag.TARGET.goldengate 1 1
2020-07-19 23:02:37: GG_HOME = /u01/oracle/goldengate/gg19_db19_MS
2020-07-19 23:02:37: GG_CONF_HOME = /mnt/acfs_rep1/deployments/ggsm19/etc/conf
2020-07-19 23:02:37: GG_VAR_HOME = /mnt/acfs_rep1/deployments/ggsm19/var
2020-07-19 23:02:37: GG_SVC_MGR_PORT = 9100
2020-07-19 23:02:37: GG_ADMIN_USERNAME = admin
2020-07-19 23:02:37: GG_OGG_CONTAINER =
2020-07-19 23:02:37: ENVIRONMENT_VARS =
2020-07-19 23:02:37: DATAGUARD_AUTOSTART = no
2020-07-19 23:02:37: VIP_NAME =
2020-07-19 23:02:37: START_TIMEOUT = 300
2020-07-19 23:02:37: executing cmd: /u01/app/19.0.0.0/grid/bin/crsctl get credmaint -path
xag_ogg_user_TARGET -credtype userpass -id 0 -attr passwd
2020-07-19 23:02:37: executing cmd: /u01/app/19.0.0.0/grid/bin/crsctl status res
xag.TARGET.goldengate -f
2020-07-19 23:02:38: Exported GGS_HOME /u01/oracle/goldengate/gg19_db19_MS
2020-07-19 23:02:38: Exported OGG_CONF_HOME /mnt/acfs_rep1/deployments/ggsm19/etc/conf
2020-07-19 23:02:38: Exported LD_LIBRARY_PATH
/u01/oracle/goldengate/gg19_db19_MS:/u01/app/19.0.0.0/grid/lib:/etc/ORCLcluster/lib
2020-07-19 23:02:38: Exported LD_LIBRARY_PATH_64 /u01/oracle/goldengate/gg19_db19_MS
2020-07-19 23:02:38: Exported LIBPATH /u01/oracle/goldengate/gg19_db19_MS
2020-07-19 23:02:38: ogg input =
{"oggHome":"/u01/oracle/goldengate/gg19_db19_MS","serviceManager":{"oggConfHome":"/mnt/acfs_rep1/de
ployments/ggsm19/etc/conf","portNumber":9100},"username":"","credential":""}
```

```
2020-07-19 23:02:38: About to exec /u01/oracle/goldengate/gg19_db19_MS/bin/XAGTask HealthCheck
2020-07-19 23:02:39: XAGTask retcode = 0
```

3. CRS trace file

Location: /u01/app/oracle/diag/crs/<hostname>/crs/trace

Example location: /u01/app/oracle/diag/crs/pnode1/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:

```
2020-07-16 20:26:49.152 : AGFW:2208286464: [ INFO] {1:9722:45224} Agent received the
message: RESOURCE_START[xag.TARGET.goldengate 1 1] ID 4098:299415

2020-07-16 20:26:49.152 : AGFW:2208286464: [ INFO] {1:9722:45224} Preparing START command
for: xag.TARGET.goldengate 1 1

2020-07-16 20:26:49.152 : AGFW:2208286464: [ INFO] {1:9722:45224} xag.TARGET.goldengate 1 1
state changed from: OFFLINE to: STARTING

2020-07-16 20:26:49.153 :CLSDYNAM:2197780224: [xag.TARGET.goldengate]{1:9722:45224} [start]
Executing action script: /u01/oracle/XAG/bin/aggoldengatescaas[start]

2020-07-16 20:26:49.305 :CLSDYNAM:2197780224: [xag.TARGET.goldengate]{1:9722:45224} [start] GG
agent running command 'start' on xag.TARGET.goldengate

2020-07-16 20:26:49.605 :CLSDYNAM:2197780224: [xag.TARGET.goldengate]{1:9722:45224} [start]
Starting OGG SCA instance

2020-07-16 20:26:52.661 :CLSDYNAM:2197780224: [xag.TARGET.goldengate]{1:9722:45224} [start]
ServiceManager fork pid = 258069

2020-07-16 20:26:52.661 :CLSDYNAM:2197780224: [xag.TARGET.goldengate]{1:9722:45224} [start] Waiting
for /mnt/acfs_repl/deployments/ggsm19/var/run/ServiceManager.pid

2020-07-16 20:26:52.661 :CLSDYNAM:2197780224: [xag.TARGET.goldengate]{1:9722:45224} [start] Waiting
for SM to start

2020-07-16 20:26:52.661 :CLSDYNAM:2197780224: [xag.TARGET.goldengate]{1:9722:45224} [start]
ServiceManager PID = 258071

2020-07-16 20:26:52.762 :CLSDYNAM:2197780224: [xag.TARGET.goldengate]{1:9722:45224} [start] execute
XAGTask HealthCheck

2020-07-16 20:26:53.814 :CLSDYNAM:2197780224: [xag.TARGET.goldengate]{1:9722:45224} [start] XAGTask
retcode = 1

2020-07-16 20:26:53.814 :CLSDYNAM:2197780224: [xag.TARGET.goldengate]{1:9722:45224} [start] XAG
HealthCheck after start returned 1

2020-07-16 20:26:53.814 : AGFW:2197780224: [ INFO] {1:9722:45224} Command: start for
resource: xag.TARGET.goldengate 1 1 completed with status: SUCCESS
```

4. GoldenGate deployment log files

Location: <Goldengate_deployment_directory>/<deployment_name>/var/log

Example location: /mnt/acfs_rep1/deployments/TARGET/var/log

File names: adminsrvr.log, recvsrvr.log, pmsrvr.log, distsrvr.log

Contains the output of start, stop and status checks of the GoldenGate deployment processes (Administration Server, Distribution Server, Receiver Server and the Performance Metrics Server).

5. GoldenGate report files

Location: <Goldengate_deployment_directory>/<deployment_name>/var/lib/report

Example location: /mnt/acfs_rep1/deployments/TARGET/var/lib/report

The GoldenGate report files contain important information, warning messages and errors for all of the GoldenGate processes, including the Manager processes. If any of the GoldenGate processes fail to start or abend when they are running, the process report file will contain important information which can be used to determine the cause of the failure.

CONNECT WITH US

Call +1.800.ORACLE1 or visit oracle.com.

Outside North America, find your local office at oracle.com/contact.

 blogs.oracle.com

 facebook.com/oracle

twitter.com/oracle

Copyright © 2020, Oracle and/or its affiliates. All rights reserved. This document is provided for information purposes only, and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document, and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group. 0120

Oracle Maximum Availability Architecture (MAA) GoldenGate Hub
December, 2020

Author: Stephan Haisley

