An Oracle White Paper
April 2011

Oracle GoldenGate high availability using Oracle Clusterware

Executive Overview 1
High Availability for Oracle GoldenGate 2
Oracle GoldenGate processes 2
Single server high availability 3
Cluster high availability 4
Oracle GoldenGate cluster high availability prerequisites 4
Shared Storage 5
About this best practice document 8
Oracle GoldenGate with Oracle Clusterware 9
About Oracle Clusterware 9
Oracle Clusterware Configuration 9
Examples 13
Oracle Clusterware 10g Release 2 and 11g Release 1 13
Oracle Clusterware 11g Release 2 22
Conclusion 29
Appendix 1: Action Script 30
Appendix 2: Extended Action Script for Oracle ASM 36

Executive Overview

Oracle GoldenGate is often used in mission-critical systems with stringent high availability requirements. In an Oracle to Oracle scenario, minimal latency between the source and the destination databases is important to achieve minimal downtime and minimal data loss in case of a failover. In an Oracle Real Application Cluster (RAC) configuration access to the database is not dependent on the availability of any one of the servers in the cluster. Oracle GoldenGate however could be impacted by non-availability of a single server since many critical components run only on one of the cluster servers.

This paper addresses how to achieve high availability for Oracle GoldenGate in a cluster configuration using Oracle Clusterware. Oracle Clusterware will ensure that Oracle GoldenGate can tolerate server failures by moving processing to another available server in the cluster. As a result Oracle GoldenGate processing is dependent on database availability rather than server availability thereby ensuring minimal latency.

This paper includes sample code that can be used to configure Oracle Clusterware to manage Oracle GoldenGate manager. The example is generic and can serve as a starting point for a more customized Oracle GoldenGate high availability implementation.

Please refer to Oracle Support note 790189.1. The example in this document falls under the category “Using Oracle Clusterware to protect any kind of application”.

High Availability for Oracle GoldenGate

In a typical production environment the following Oracle GoldenGate processes will be running (also see Figure 1 below):

- Manager at source and target systems (always). The manager process overlooks other Oracle GoldenGate processes.
- Zero or more extract processes on the source system to capture from the transaction logs.
- Zero or more extract processes on the source system to send data to one or more target systems. In Oracle GoldenGate terminology these processes are called data pumps.
- Zero or more replicat processes on the target system to apply changes from a trail to a target database.
- Zero or more server collector processes on the target system that receive trails sent by a pump (extract) running on another server.

Oracle GoldenGate processes

Figure 1 below shows a high level overview of the Oracle GoldenGate processes for a typical uni-directional Oracle source to Oracle target replication scenario. The boxes indicate high level source and target configuration without going into details on the actual implementation, although Figure 1 does assume the source configuration is an Oracle RAC database. For additional information about numbers in Figure 1 refer to the legend on the next page. The numbers are not meant to indicate any ordering.

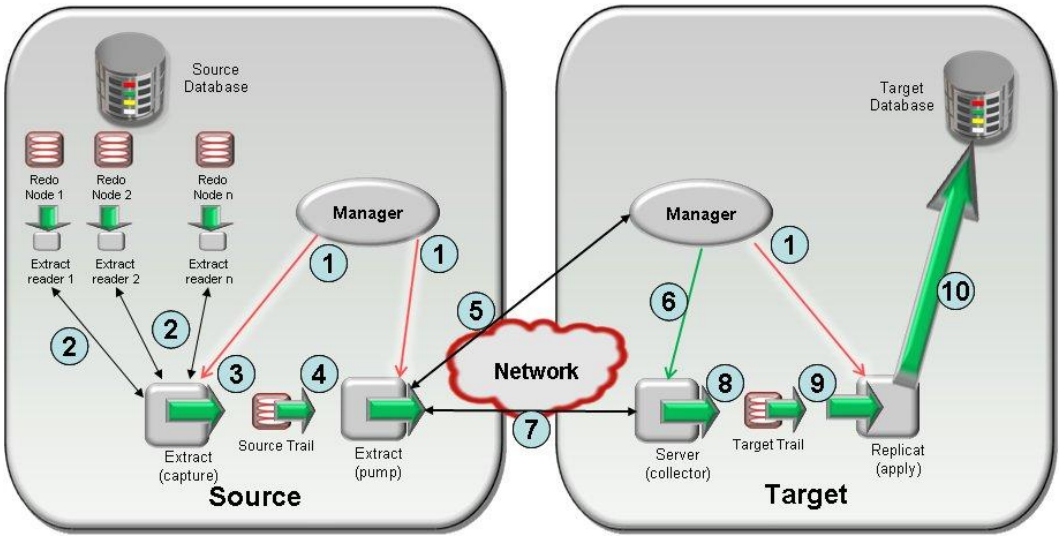

Figure 1. Oracle GoldenGate processes

Legend to Figure 1:

1. Oracle GoldenGate manager starts extract (capture/pump) and replicat (apply) processes.
2. Extract (capture) spawns an extract reader thread for every node on the source database. The coordinator thread receives, interprets and orders redo from all readers.
3. Extract (capture) writes the source trail in transaction commit order.
4. Extract (pump) reads the source trail as transactions are written to the trail.
5. Extract (pump) contacts the manager process on the target system over TCP/IP to initialize trail transfer. Manager indicates to extract which port to use to send the trail.
6. Manager on the target spawns a server collector process on a TCP/IP port.
7. After the initialization with manager extract (pump) sends trails directly to server (collector).
8. Server (collector) writes the data it receives to a local (target) trail.
9. Replicat (apply) reads the target trail as it arrives.
10. Replicat (apply) applies the transactions to the target database using the Oracle Call Interface (OCI) and/or SQL.

Single server high availability

Oracle GoldenGate's single server configuration high availability approach is achieved through configuration parameters in the manager process. The manager parameters `AUTOSTART` and `AUTORESTART` attempt to prevent Oracle GoldenGate outages and ensure Oracle GoldenGate processes get restarted so long as the manager process is running. Oracle GoldenGate manager starts server collector processes automatically when it receives a request from a remote extract to send trails.

Certain failures beyond Oracle GoldenGate may cause Oracle GoldenGate outages. E.g.

- Network failure.
- Database crashes or problems (e.g. database (or database instance) is down).
- Data integrity issues (e.g. problems caused by an out-of-sync condition).
- Server issues (e.g. OS crash).

In the event that the above outages result in a failure of the manager process, the manager process must be restarted. It is important to monitor the manager process since in some cases, operator intervention could be required. Management Pack for Oracle GoldenGate provides

capabilities to set up notifications such as email alerts etc. Third-party products can also be used for monitoring.

Cluster high availability

An Oracle GoldenGate configuration in a cluster builds on top of the single server configuration. The approaches you would use in a single server configuration to keep Oracle GoldenGate extract and replicat available, using `AUTOSTART` and `AUTORESTART` parameters in the manager parameter file, still apply.

Oracle GoldenGate cluster high availability prerequisites

In a cluster configuration – on the source and/or on the target – Oracle GoldenGate runs on a single server at any time. If that server goes down (e.g. due to system maintenance, a crash, etc.) then Oracle GoldenGate can be started on another server. In order for Oracle GoldenGate to resume processing, you must, at a minimum, configure the recovery-related[1] Oracle GoldenGate files in a centrally shared location:

- checkpoint files (`$GGATE_HOME/dirchk`)
- trail files

Additionally the parameter files (`$GGATE_HOME/dirprm`) will have to be identical between different nodes in the cluster[2]. You may choose to share these amongst all Oracle GoldenGate installations on the different nodes also.

The shared files should be stored on shared storage that is available independent of any particular server availability. I.e. a shared local directory (e.g. mounted on other servers using NFS) is not recommended since the server hosting the shared directory may not be available at all times. Make sure that your file system fulfills your availability and performance requirements.

You may choose to install Oracle GoldenGate in a central location, for example on a cluster file system, so that every server can run Oracle GoldenGate from this single installation. Of course Oracle GoldenGate cannot run on multiple servers at any point in time.

---

[1] The term "recovery-related" files will be used in the context of Oracle GoldenGate throughout this document to denote the checkpoint files and trail files.

[2] Environment settings may be different from one node to the other but you should resolve these differences either through environment settings that are inherited from the parent process or by using a reference to a node-specific Oracle GoldenGate macro file so that the actual parameter files can always be identical between the nodes in the cluster.

Shared Storage

Most shared storage solutions, including general purpose cluster file systems, can be used to install Oracle GoldenGate or to store the files that Oracle GoldenGate needs to recover. The following options are available from Oracle at no additional cost:

- Oracle Cluster File System (OCFS2) – available only on Linux.
- Oracle Automatic Storage Management (ASM) Cluster File System (ACFS). Oracle ACFS was introduced with Oracle Database 11g Release 2.
- Oracle DataBase File System (DBFS). Oracle DBFS was introduced with Oracle Database 11g Release 2.

Oracle Cluster File System (OCFS2)

One of the options you may choose to use on Linux is Oracle Cluster File System (OCFS2) which is included in recent Linux distributions (included in the Linux kernel in some distributions). OCFS2 is an open source general purpose cluster file system. Instead of installing Oracle GoldenGate in a local directory you would install Oracle GoldenGate in a directory that is mounted to an OCFS2 volume. Refer to the OCFS2 website for more information: http://oss.oracle.com/projects/ocfs2/.

OCFS2 can also be used for Oracle Database storage, although Oracle recommends the use of Oracle Automatic Storage Management (ASM) starting with Oracle Database 10g.

Oracle ASM Cluster File System (ACFS)[3]

Oracle Database 11g Release 2 introduces the Oracle Automatic Storage Management Cluster File System (ASM Cluster File System, ACFS). ACFS is a general purpose single-node (standalone) or cluster file system on top of ASM (but outside a database). ACFS can be accessed using industry-standard Network Attached Storage (NAS) file access protocols: Network File System (NFS) and Common Internet File System (CIFS).

ACFS file systems would generally be mounted on all nodes in a cluster. As a result ACFS can be used to install Oracle GoldenGate to make it accessible to all nodes, and to store processing files required to failover between nodes in case of a failure.

[3] With the initial release of Oracle Database 11.2.0.1 ACFS is not available on all platforms. Please check the release notes to ensure your platform is supported if you want to use ACFS.

ACFS is part of ASM as part of the Oracle Database 11g Release 2 Grid Infrastructure installation. Oracle Database 11g Release 2 Clusterware and ASM must be used for Oracle Database 11g Release 2, but can also be used for Oracle Database 11g Release 1 or Oracle Database 10g Release 2.

For more information about the Oracle Database 11g Release 2 ACFS, please refer to the Oracle Database Storage Administrator's Guide as part of the Oracle Database 11g Release 2 documentation set (http://download.oracle.com/docs/cd/E11882_01/server.112/e10500/toc.htm).

Database File System (DBFS)

Oracle Database 11g Release 2 also introduces a Database File System (DBFS). In DBFS files are stored as secure files which are internally stored as LOB data values in the Oracle Database. In-database storage provides high availability, security and encryption capabilities that may not be otherwise available on general purpose file systems. In a cluster configuration the DBFS can be accessed from multiple nodes, and hence it can act as a cluster file system.

Files in DBFS can be managed through a set of PL/SQL APIs. In order to mount a DBFS as an OS file system another component, the DBFS client (dbfs_client) is required. For Oracle Database 11.2.0.1 you can only mount a DBFS file system on Linux.

When DBFS is accessed through the DBFS client it has some restrictions compared to general purpose file systems. As a result you should not perform an Oracle GoldenGate installation in DBFS. You can however store the recovery-related files in a cluster configuration in DBFS to make them accessible to all nodes. A DBFS mounted on multiple servers concurrently does not support file locking[4]. If the DBFS is mounted on one server at the time then file locking is supported.

DBFS requires an Oracle Database 11g Release 2 (or higher) database. You can use DBFS to store Oracle GoldenGate recovery-related files for lower releases of the Oracle Database, but you will have to create a separate Oracle Database 11g Release 2 (or higher) database to host the file system.

[4] Oracle GoldenGate manager uses a file lock to determine whether it should attempt to restart a process through the `AUTORESTART` parameter. Although an attempt to restart a process would probably still fail if the process is already running (even if it is running on another node) the number of restart attempts (configurable) is limited. Without file locking there may be a situation where there is a genuine process outage that could be resolved through `AUTORESTART` but the restart attempts have been exhausted. Such a scenario would cause an unnecessary Oracle GoldenGate outage.

For more information about DBFS, its restrictions as well as how to configure a DBFS, please refer to the Oracle Database SecureFile and Large Objects Developer's Guide as part of the Oracle Database 11g Release 2 documentation set (http://download.oracle.com/docs/cd/E11882_01/appdev.112/e10645/toc.htm).

About this best practice document

This best practice document will not be able to provide an out-of-the-box solution to any and all issues causing Oracle GoldenGate outages. Any configuration can be made more highly available beyond the generic example in this paper through additional implementation-specific coding.

In any and all critical implementations you have to configure notification procedures to minimize the impact of any Oracle GoldenGate outages, irrespective whether the outages are related or unrelated to Oracle GoldenGate.

The main issues that this best practice document will help address are server-related issues, to some degree database problems and in rare cases network issues. Other problems, for example data issues, but various other issues, will have to continue to be solved beyond Oracle GoldenGate, often (only) through manual intervention. Once resolved Oracle GoldenGate will resume processing where it left off, assuming the recovery-related files are available.

The implementation of the best practice in this document requires Oracle Clusterware 10g Release 2 or higher. Earlier versions of Oracle Clusterware cannot be used. The example included in this document should work on most Unix and Linux environments (it has been tested on Linux with Oracle Database 11g Release 1 and Release 2 and Oracle GoldenGate v10 and v10.4). For Windows the same approach can be used but the sample script will have to be modified to run on Windows.

Third-party cluster management software such as Veritas Cluster Server, Sun Cluster, Redhat Cluster Manager etc. likely provide similar capabilities but will not be discussed in this document. This document may still be a useful reference for other cluster management solutions but the included example will not work.

Oracle GoldenGate with Oracle Clusterware

This section provides an introduction into Oracle Clusterware and indicates how to install Oracle GoldenGate in a cluster configuration.

About Oracle Clusterware

Oracle Clusterware is Oracle's cluster management software. The software manages node membership and provides functionality such as resource management and high availability.

Starting with Oracle Clusterware 10g Release 2 Oracle Clusterware provides the capability to manage third-party applications. There are commands to register an application and instruct Oracle Clusterware how to manage the application in a clustered environment. This capability will be used to register the Oracle GoldenGate manager process as an application managed through Oracle Clusterware.

Oracle Clusterware can be installed standalone without an Oracle RAC database and still manage a cluster of servers and various applications running on these servers. As such Oracle Clusterware can also be installed on more than just the database servers to form a single cluster. For example you may use 4 database servers and 2 additional Oracle GoldenGate servers in a single cluster. The Oracle Database would only run on the 4 database servers and Oracle GoldenGate would only run on one of the two available Oracle GoldenGate servers (with failover to the other server dedicated to Oracle GoldenGate in case that server fails).

Oracle Clusterware Configuration

This section discusses how to include Oracle GoldenGate in an Oracle Clusterware configuration. It is assumed that Oracle Clusterware has already been installed and is functioning. Refer to the Oracle documentation starting at http://docs.oracle.com on how to install Oracle Clusterware.

Oracle GoldenGate installation

Oracle GoldenGate must be available on every server in the same location (e.g. `/u01/app/ggate`). You may choose to perform a local installation on every server, or a single installation on a shared file system. You will need shared storage for the recovery-related files. On a Unix/Linux platform you can use a symbolic link to a central location for the shared directories.

The environments used to validate the examples in this document used a shared OCFS2 volume and an ACFS volume for the Oracle GoldenGate installation.

Virtual IP address (VIP)

Oracle Clusterware uses the concept of a Virtual IP address (VIP) to manage high availability for applications that require incoming network traffic (including the Oracle RAC database). A VIP is an IP address on the public subnet that can be used to access a server. If the server hosting the VIP were to go down, then Oracle Clusterware will migrate the VIP to a surviving server to minimize interruptions for the application accessing the server (through the VIP). This concept enables faster failovers compared to time-out based failovers on a server's actual IP address in case of a server failure.

For Oracle GoldenGate, you should use a VIP to access the manager process to isolate access to the manager process from the physical server that is running Oracle GoldenGate. Remote pumps must use the VIP to contact the Oracle GoldenGate manager. The Management Pack for Oracle GoldenGate, if used, should use the VIP to contact the Oracle GoldenGate manager. The VIP must be an available IP address on the public subnet and cannot be determined through DHCP. Ask a system administrator for an available fixed IP address for Oracle GoldenGate managed through Oracle Clusterware.

START, CHECK, STOP, CLEAN and ABORT routines

Oracle Clusterware must be instructed how to start the program, check whether it is running, and stop it. Programs using various programming languages, including shell scripts, can be registered and run by Oracle Clusterware.

START

Oracle GoldenGate manager is the process that starts all other processes Oracle GoldenGate processes. The only process that Oracle Clusterware should start is the manager process. Use the `AUTOSTART` parameter in the manager parameter file to start extract and replicat processes. You can use wild cards (`AUTOSTART ER *`) to start all extract and replicat processes when manager is started, but note that any initial load extract and/or replicats will start with the unlimited `ER *` wild card. Based on a naming convention you use, you may use more restrictive wild cards (e.g. `AUTOSTART EXTRACT cdc*`) or list specific extracts/replicats you want to start automatically. Refer to the Oracle GoldenGate Reference Guide for the details on `AUTOSTART`).

Also note that once manager is started through Oracle Clusterware, it is Oracle Clusterware that manages its availability. If you would stop manager through the command interface ggsci, then Oracle Clusterware will attempt to restart it. Use the Oracle Clusterware commands (see the example in the next chapter) to stop Oracle GoldenGate and prevent Oracle Clusterware from attempting to restart it.

All Oracle GoldenGate processes except manager can still be controlled through ggsci. In the recommended setup, Oracle Clusterware will not interfere with ggsci commands that manipulate

Oracle GoldenGate processes. In most production environments however you probably want all processes to be running all the time. Use `AUTORESTART` in the manager parameter file for manager to automatically attempt to restart any processes that would go down. Also, make sure to have the necessary notification procedures in place if processes were to go down and stay down for some reason.

CHECK

The validation whether Oracle GoldenGate is running is equivalent to making sure the Oracle GoldenGate manager runs. Use the `AUTORESTART` parameter in the manager parameter file to ensure that extract and replicat processes will be restarted if/when they go down. Also make sure to have a notification infrastructure in place to prevent Oracle GoldenGate processes from staying down for an extended period of time due to errors that are beyond Oracle GoldenGate's control (e.g. data errors).

You may choose to implement very extensive checking to ensure all Oracle GoldenGate processes are running fine but it does not make much sense to let Oracle Clusterware manage any other processes but Oracle GoldenGate manager. The only reason why Oracle Clusterware may be able to start a process when Oracle GoldenGate manager cannot, would be related to the environment settings such as the `ORACLE_HOME` setting. These settings ought to be corrected so that Oracle GoldenGate manager can always start its processes.

STOP

Stop must stop all Oracle GoldenGate processes, including manager. Stop may be called during a planned downtime (e.g. a server is taken out of a cluster for maintenance reasons) and/or if you manually instruct Oracle Clusterware to relocate Oracle GoldenGate to a different server (e.g. to change the load on a server). If a server crashes then all processes will go down with it, in which case they can be started on another server.

CLEAN

Clean was introduced with Oracle Clusterware 11g Release 2. It will not be used for Oracle Clusterware 10g Release 2 or 11g Release 1. Clean is called when there is a need to clean up the resource. It is a non-graceful operation.

ABORT

Abort was introduced with Oracle Clusterware 11g Release 2. It will not be used for Oracle Clusterware 10g Release 2 or 11g Release 1. Abort is called if any of the resource components hang to abort the ongoing action. Abort is not required to be included.

Program registration, start and stop

Once Oracle GoldenGate has been installed across the cluster and a script to start, check and stop (and for Oracle Clusterware 11g Release 2, clean and optionally abort) has been written and has been made accessible to all nodes, Oracle GoldenGate can be registered in Oracle Clusterware. Use the Clusterware commands to create, register and set privileges on the VIP and the Oracle GoldenGate application. Once registered, use the Oracle Clusterware commands to start, relocate and stop Oracle GoldenGate. For detailed steps see the example in the next section.

The Oracle Clusterware commands are documented in an appendix in the Oracle Clusterware Administration and Deployment Guide:

- Oracle Database 10g Release 2: http://download.oracle.com/docs/cd/B19306_01/rac.102/b14197/crsref.htm#CHEIIJHE
- Oracle Database 11g Release 1: http://download.oracle.com/docs/cd/B28359_01/rac.111/b28255/crsref.htm#CHEIIJHE
- Oracle Database 11g Release 2: http://download.oracle.com/docs/cd/E11882_01/rac.112/e10717/crsref.htm#CHDGADEH

Examples

This section goes step-by-step through a couple of examples. This section covers a separate example for Oracle Clusterware 10g Release 2 and 11g Release 1 versus Oracle Clusterware 11g Release 2. Please refer to the relevant section below depending on which Oracle Clusterware release you use.

Oracle Clusterware 10g Release 2 and 11g Release 1

This section provides an example for Oracle Clusterware 10g Release 2 and 11g Release 1.

Step 1: Add an application VIP

The first step is to create an application VIP. The VIP will be used to access Oracle GoldenGate (e.g. by a remote pump or by the Management Pack for Oracle GoldenGate). Oracle Clusterware will assign the VIP to a physical server, and migrate the VIP if that server were to go down or if you instruct Clusterware to do so.

To create the application VIP, login as the OS Oracle software owner (`oracle` in this example) and run:

```
CLUSTERWARE_HOME/bin/crs_profile -create ggatevip \
       -t application \
       -a CLUSTERWARE_HOME/bin/usrvip \
       -o oi=eth0,ov=192.168.1.23,on=255.255.255.0
```

with:

- `CLUSTERWARE_HOME` as the oracle home in which Oracle Clusterware is installed (e.g. `/u01/app/oracle/crs111`).
- `ggatevip` is the name of the application VIP that you will create.
- `oi=eth0`; `eth0` is the public interface in this example.
- `ov=192.168.1.23`; the virtual IP address is `192.168.1.23` in this example.
- `on=255.255.255.0`; the subnet mask. This should be the same subnet mask for the public (general) IP address.

There are more options you can set through the `crs_*` commands. For example, you can indicate what nodes can be used to host the application, if there is a preference for a node to run the application, and whether you want to always start the application upon reboot, never, or restore the last state upon reboot (the default). Please refer to the Oracle Clusterware documentation for details:

Oracle Clusterware 10g Release 2:
http://download.oracle.com/docs/cd/B19306_01/rac.102/b14197/crsref.htm#i1018873;
Oracle Clusterware 11g Release 1:
http://download.oracle.com/docs/cd/B28359_01/rac.111/b28255/crsref.htm#i1018873.

Next, register the VIP as `oracle`:

```
CLUSTERWARE_HOME/bin/crs_register ggatevip
```

Because the assignment of an IP address is done by the `root` user, you have to set the ownership of the VIP to the `root` user. Connect as `root` and execute:

```
CLUSTERWARE_HOME/bin/crs_setperm ggatevip -o root
```

As `root`, allow `oracle` to run the script to start the VIP.

```
CLUSTERWARE_HOME/bin/crs_setperm ggatevip -u user:oracle:r-x
```

Then, as `oracle`, start the VIP:

```
CLUSTERWARE_HOME/bin/crs_start ggatevip
```

To validate whether the VIP is running and on which node it is running, execute:

```
CLSUTERWARE_HOME/bin/crs_stat ggatevip -t
```

For example:

```
[oracle@rac2 bin]$ crs_stat ggatevip -t
Name           Type           Target    State     Host
------------------------------------------------------------
ggatevip       application    ONLINE    ONLINE    rac2
```

At this point you can also connect to another server in the subnet and ping the VIP's IP address. You should get a reply from this IP address.

```
mvandewiel@MVDW-LT-01:~$ ping -c4 192.168.1.23
PING 192.168.1.23 (192.168.1.23) 56(84) bytes of data.
64 bytes from 192.168.1.23: icmp_seq=1 ttl=64 time=2.22 ms
64 bytes from 192.168.1.23: icmp_seq=2 ttl=64 time=0.665 ms
64 bytes from 192.168.1.23: icmp_seq=3 ttl=64 time=0.095 ms
64 bytes from 192.168.1.23: icmp_seq=4 ttl=64 time=0.209 ms

--- 192.168.1.23 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3002ms
rtt min/avg/max/mdev = 0.095/0.798/2.225/0.851 ms
```

Step 2: Create an action program

The action program must be able to accept 3 parameter values: start, stop or check.

- start and stop: returns 0 if successful, 1 if unsuccessful.
- check: returns 0 if Oracle GoldenGate is running, 1 if it is not running.

Please refer to Appendix 1 for an example action script. The script listed in the appendix also accepts the clean and abort arguments that are not used when you use Oracle Clusterware 10g Release 2 or 11g Release 1.

Note that the generic script will verify whether Oracle GoldenGate manager is running and if it is, assume that all is fine (remember that the manager parameter file contains `AUTOSTART` and `AUTORESTART` parameters similar to a single instance installation of Oracle GoldenGate). The action script can easily be extended to check for specific extract and/or replicat processes, to ensure that they are running (and optionally send out a notification if they are not). However Oracle Clusterware should not be used to start these processes explicitly.

Save the script in a file `goldengate_action.scr` and copy the file to every node in the cluster into the same directory (or, if you use shared storage, you can store the script on shared storage). The remainder of this example assumes that you stored the script in the default Oracle Clusterware location `CLUSTERWARE_HOME/crs/public`. Make sure the script is executable by the Oracle GoldenGate software owner (`chmod +x goldengate_action.scr`).

Step 3: Create an application profile

The application profile is a text file with key-value pairs. Rather than creating such a text file from scratch you use `CLUSTERWARE_HOME/bin/crs_profile` to create and manipulate the file. Connect as `oracle` and execute:

```
CLUSTERWARE_HOME/bin/crs_profile \
       -create goldengate_app \
       -t application \
       -r ggatevip \
       -a CLUSTERWARE_HOME/crs/public/goldengate_action.scr \
       -o ci=10
```

The following values are used:

- `-create goldengate_app`: the application name is `goldengate_app`.
- `-r ggatevip`: -r specifies the required resources that must be running for the application to start. In this case the VIP `ggatevip` must be running before Oracle GoldenGate starts.

- `-a CLUSTERWARE_HOME/crs/public/goldengate_action.scr` specifies the action script (reminder: this script must be available in this location on every server and be executable).
- `-o ci=10`: the -o flag specifies options. In this case the only option that is specified is the Check Interval which is set to 10 seconds.

For more information about the `crs_profile` command and its options, please refer to the Oracle Clusterware documentation.

The next step is to register the application with Oracle Clusterware. Run this command as `oracle`:

```
CLUSTERWARE_HOME/bin/crs_register goldengate_app
```

Unless you installed Oracle GoldenGate as Oracle software owner, you will have to change the ownership of the application to your Oracle GoldenGate software owner. This example assumes you use `ggate` as the Oracle GoldenGate software owner. If you use `oracle` as the Oracle GoldenGate software owner then you can skip the following 2 commands.

Login as `root` and change the ownership of the `goldengate_app` application:

```
CLUSTERWARE_HOME/bin/crs_setperm goldengate_app -o ggate
```

As `root`, allow `oracle` to run the script to start the `goldengate_app` application.

```
CLUSTERWARE_HOME/bin/crs_setperm goldengate_app -u user:oracle:r-x
```

Step 4: Start the application

From now on you should always use Oracle Clusterware to start Oracle GoldenGate. Login as `oracle` and execute:

```
CLUSTERWARE_HOME/bin/crs_start goldengate_app
```

To check the status of the application:

```
CLUSTERWARE_HOME/bin/crs_stat goldengate_app -t
```

For example:

```
[oracle@rac1 bin]$ crs_stat goldengate_app -t
Name           Type           Target    State     Host
------------------------------------------------------------
goldengate_app application    ONLINE    ONLINE    rac2
[oracle@rac1 bin]$
```

Manage the application

When Oracle GoldenGate is running, and you want to move Oracle GoldenGate to run on a different server, you can use the `CLUSTERWARE_HOME/bin/crs_relocate` command with the force option to move the VIP as well (as `oracle`, on any node):

```
[oracle@rac1 bin]$ crs_relocate -f goldengate_app
Attempting to stop `goldengate_app` on member `rac2`
Stop of `goldengate_app` on member `rac2` succeeded.
Attempting to stop `ggatevip` on member `rac2`
Stop of `ggatevip` on member `rac2` succeeded.
Attempting to start `ggatevip` on member `rac1`
Start of `ggatevip` on member `rac1` succeeded.
Attempting to start `goldengate_app` on member `rac1`
Start of `goldengate_app` on member `rac1` succeeded.
[oracle@rac1 bin]$
```

Application relocation is exactly what happens when the server running Oracle GoldenGate crashes.

To stop Oracle GoldenGate, use (as `oracle`):

```
CLUSTERWARE_HOME/bin/crs_stop goldengate_app
```

Cleanup

If you want to stop Oracle Clusterware from managing Oracle GoldenGate, and you want to cleanup the changes you made, then:

Stop Oracle GoldenGate (login as `oracle`):

```
CLUSTERWARE_HOME/bin/crs_stop goldengate_app
```

Stop the VIP (as `oracle`):

```
CLUSTERWARE_HOME/bin/crs_stop ggatevip
```

Unregister the application `goldengate_app` (login as `ggate`):

```
CLUSTERWARE_HOME/bin/crs_unregister goldengate_app
```

Unregister the VIP (login as `root`):

```
CLUSTERWARE_HOME/bin/crs_unregister ggatevip
```

Delete the `goldengate_app` profile (as `oracle`; on the node where you created the profile):

```
CLUSTERWARE_HOME/bin/crs_profile -delete goldengate_app
```

Delete the VIP profile (as `oracle`; on the node where you created the profile):

```
CLUSTERWARE_HOME/bin/crs_profile -delete ggatevip
```

Notes on using Oracle ASM for Oracle GoldenGate extract

This section is only relevant if you extract out of an Oracle RAC database that uses Oracle Automatic Storage Management (ASM) to store its redo and/or archive logs. This section only applies to Oracle ASM 10g Release 2 or 11g Release 1. If you use Oracle ASM 11g Release 2 (possibly with a lower version of the Oracle Database) then please refer to the section Notes on using Oracle Automatic Storage Manager (ASM) 11g Release 2.

Oracle GoldenGate supports data capture from an Oracle Database using ASM. A few additional setup steps are required in order to use ASM for Oracle redo and/or archive logs:

- Extract requires a connection into an ASM instance to be able to read the transaction logs. The connection has to go through the Oracle Database listener and because the ASM instance is only mounted (not open) an entry for the ASM instance must be added to the listener configuration file in order to let incoming connections go through. See the Oracle GoldenGate for Windows and Unix Administrator Guide as well as Oracle Support note 340277.1 for more details.
- The tnsnames.ora file(s) or LDAP directory must include entries to the ASM instance(s) in order for connect strings to resolve connection requests.
- The Oracle GoldenGate extract parameter file must include the following line:
- `TRANLOGOPTIONS ASMUSER <user>@<asm>, ASMPASSWORD <password>, ENCRYPTKEY <key>`

An ASM instance is not a regular database instance and does not support the concept of regular database users. As a result the user Oracle GoldenGate uses to connect to ASM is always an administrative user (`SYSDBA` or `SYSASM`) which would enable startup and shutdown of the instance. The connection attempt to the Oracle ASM instance will not fail if the instance is down, but Oracle GoldenGate extract will not be able to start because any queries issued against the ASM instance will fail.

If you use Oracle GoldenGate to extract from an Oracle RAC database that uses ASM to store its logs, and your connection to ASM could be routed to any ASM instance in the cluster, then you may run into the situation that extract connects to an ASM instance that happens to be down. In that case extract will abend, although the database may still be running fine on other servers. This will cause the extract process to fall behind.

The following extended example will prevent this case from happening. The steps to configure high availability for Oracle GoldenGate using Oracle Clusterware remain the same as discussed in earlier sections in this paper. Also, all configurations that were discussed earlier, including

running Oracle GoldenGate on its own server with failover to a database server in case the Oracle GoldenGate server fails, still work. The action script has been extended to ensure Oracle GoldenGate extract will always connect to an ASM instance that is up and running.

You can use this generic example for your setup or use and modify it for your own optimal implementation.

Assumptions and prerequisites

The example action script listed in Appendix 2 works under the following assumptions:

- Every node in the cluster includes an entry in its `listener.ora` configuration file to connect to the ASM instance that is managed by that node. Below is an example of a `listener.ora` file for one node in the cluster. The bold section has been added to enable connections into the local ASM instance:

```
LISTENER_RAC1 =
 (DESCRIPTION_LIST =
 (DESCRIPTION =
 (ADDRESS = (PROTOCOL = TCP)(HOST = rac1-vip)(PORT = 1521))
    )
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.1.11)(PORT = 1521))
    )
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC))
    )
  )

SID_LIST_LISTENER_RAC1 =
  (SID_LIST =
    (SID_DESC =
      (GLOBAL_DBNAME = ASM)
      (ORACLE_HOME = /u01/oracle/ora111)
      (SID_NAME = +ASM1)
    )
  )
```

  Note that by using `GLOBAL_DBNAME = ASM` implicitly Oracle Clusterware will create a service ASM which can referenced in the `tnsnames.ora` file.

- The shared `tnsnames.ora` file or the `tnsnames.ora` file on every node (or the LDAP directory if one is used) contains specific connect information for every ASM instance. The

name that is used for the connect entry must match the name that Oracle Clusterware uses to identify the ASM instance (in this example ASM1 and ASM2):

```
ASM1 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = rac1-vip)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = ASM)
      (INSTANCE_NAME = +ASM1)
    )
  )

ASM2 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = rac2-vip)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = ASM)
      (INSTANCE_NAME = +ASM2)
    )
  )
```

Note that `ASM1` connects to the ASM instance on node 1, `ASM2` to node 2, etc. Match your configuration with your ASM setup.

- Oracle GoldenGate extract uses a library reference (using the parameter `INCLUDE`) to connect to the ASM instance. This macro file will be created/maintained by the action script in Appendix 2 using status information for the ASM instances obtained through Oracle Clusterware commands. Use your own extract parameter file but replace the entry to connect to the ASM instance with the `INCLUDE` reference to the library. Below is an example of an extract parameter file with the library `INCLUDE` highlighted in bold.

```
extract esrc
userid src@racdb, password asdf786sq, encryptkey mykey

#include an asm connection library
#ensure this line matches the action script
include ./dirprm/asmconn.lib

exttrail ./dirdat/aa
```

```
reportcount every 1 minute, rate

table src.* ;
```

- An extended action script (see following section and Appendix 2) is used.

Modified action script for ASM connectivity

The modified action script in Appendix 2 includes a routine that will (re)create the ASM connection library. Make sure the name and location of this library match your extract parameter file(s). Also, ensure you use the correct password and encryption key.

Note that the connection library will be recreated every time a start or a check call is processed. The default setting for `CHECK_INTERVAL` in Oracle Clusterware is 60 seconds which can be modified through the Oracle Clusterware commands. If extract would be running and the active ASM instance it is connected to goes down then upon restart it is assumed that the ASM connection library will have been updated to reflect the current status of the ASM instances and hence a different connect string will be used. You can modify the `CHECK_INTERVAL` (in seconds) to increase/decrease the calls to the check routine. See the Oracle Clusterware Administration and Deployment Guide for more details. Also, note that the `AUTORESTART` parameter in the Oracle GoldenGate manager parameter file has a `WAITMINUTES` attribute. Set `WAITMINUTES` (in minutes) to a higher interval than the `CHECK_INTERVAL` (in seconds) in Oracle Clusterware to ensure the ASM connection library will have been recreated after extract abends.

The modified action script with the ASM connection routine and routine calls highlighted in bold is included in Appendix 2.

Oracle Clusterware 11g Release 2

This section provides an example for Oracle Clusterware 11g Release 2.

Step 1: Add an application VIP

The first step is to create an application VIP. The VIP will be used to access Oracle GoldenGate (e.g. by a remote pump or by the Management Pack for Oracle GoldenGate). Oracle Clusterware will assign the VIP to a physical server, and migrate the VIP if that server were to go down or if you instruct Clusterware to do so.

To create the application VIP, login as `root` and run:

```
GRID_HOME/bin/appvipcfg create -network=1 \
       -ip=10.1.41.93 \
       -vipname=mvggatevip \
       -user=root
```

with:

- `GRID_HOME` as the oracle home in which Oracle 11g Release 2 Grid infrastructure components have been installed (e.g. `/u01/app/grid`).
- `-network` refers to the network number that you want to use. With Oracle Clusterware 11.2.0.1 you can find the network number using the command:

  ```
  crsctl stat res -p |grep -ie .network -ie subnet |grep -ie name -ie subnet
  ```

  Sample output is:

  ```
  NAME=ora.net1.network
  USR_ORA_SUBNET=10.1.41.0
  ```

  `net1` in `NAME=ora.net1.network` indicates this is network 1, and the second line indicates the subnet on which the VIP will be created.
- `mvggatevip` is the name of the application VIP that you will create

.

Oracle recommends the use of the `appvipcfg` utility to define applications VIPs. The VIP is created with a set of pre-defined settings and dependencies. Please refer to the Oracle Clusterware documentation for further details:

http://download.oracle.com/docs/cd/E11882_01/rac.112/e10717/crschp.htm#BGBJHJHC

As `root`, allow the Oracle Grid infrastructure software owner (e.g. `oracle`) to run the script to start the VIP.

```
GRID_HOME/bin/crsctl setperm resource mvggatevip -u user:oracle:r-x
```

Then, as `oracle`, start the VIP:

```
GRID_HOME/bin/crsctl start resource mvggatevip
```

To validate whether the VIP is running and on which node it is running, execute:

```
GRID_HOME/bin/crsctl status resource mvggatevip
```

For example:

```
[oracle@coe-01 ~]$ crsctl status resource mvggatevip
NAME=mvggatevip
TYPE=app.appvip.type
TARGET=ONLINE
STATE=ONLINE on coe-02
```

At this point you can also connect to another server in the subnet and ping the VIP's IP address. You should get a reply from this IP address.

```
mvandewiel@MVDW-LT-01:~$ ping -c4 mvggatevip
PING mvggatevip.goldengate.com (10.1.41.93) 56(84) bytes of data.
64 bytes from mvggatevip.goldengate.com (10.1.41.93): icmp_seq=1 ttl=62 time=0.378 ms
64 bytes from mvggatevip.goldengate.com (10.1.41.93): icmp_seq=2 ttl=62 time=0.511 ms
64 bytes from mvggatevip.goldengate.com (10.1.41.93): icmp_seq=3 ttl=62 time=0.426 ms
64 bytes from mvggatevip.goldengate.com (10.1.41.93): icmp_seq=4 ttl=62 time=0.433 ms

--- mvggatevip.goldengate.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3002ms
rtt min/avg/max/mdev = 0.378/0.437/0.511/0.047 ms
```

Step 2: Develop an agent script

Oracle Clusterware runs resource-specific commands through an entity called an agent. The agent script must be able to accept 5 parameter values: start, stop, check, clean and abort (optional).

Appendix 1 provides a sample agent. The agent script in Appendix 1 will also mount/unmount a DBFS file system upon startup/failover. If you do not use DBFS for recovery-related files then you can remove the calls to the DBFS mount and unmount functions.

Note that this generic agent action script will verify whether manager is running and if it is, assume that all is fine (remember that the manager parameter file contains AUTOSTART and AUTORESTART parameters similar to a single instance installation of Oracle GoldenGate). The agent action script can easily be extended to check for specific extract and/or replicat processes, to ensure that they are running (and optionally send out a notification if they are not). However for a generic implementation Oracle Clusterware should not be used to start these processes explicitly.

Save the script in a file 11gr2_gg_action.scr and copy it to every node in the cluster into the same directory (or, if you use shared storage such as OCFS2 or ACFS, you can store the script on shared storage assuming the shared file system is mounted at the same mount point on all servers). The remainder of this example assumes that you stored the script in /mnt/acfs/oracle/grid. Make sure the script is executable by the Oracle GoldenGate software owner (chmod +x 11gr2_gg_action.scr).

Step 3: Register a resource in Oracle Clusterware

Use the GRID_HOME/bin/crsctl utility to register Oracle GoldenGate as a resource in Oracle Clusterware.

Connect as the oracle and execute:

```
GRID_HOME/bin/crsctl add resource ggateapp \
     -type cluster_resource \
     -attr  "ACTION_SCRIPT=/mnt/acfs/oracle/grid/11gr2_gg_action.scr,
CHECK_INTERVAL=30, START_DEPENDENCIES='hard(mvggatevip)
pullup(mvggatevip)', STOP_DEPENDENCIES='hard(mvggatevip)'"
```

The following values are used:

- ggateapp: the resource name is ggateapp.
- START_DEPENDENCIES: there is both a hard and a startup dependency on mvggatevip. This indicates that the VIP and the ggateapp application should always start together.
- STOP_DEPENDENCIES: there is a hard stop dependency on mvggatevip. This indicates that the VIP and the ggateapp application should always stop together.

Please note that this example assumes that all servers in the cluster can host the Oracle GoldenGate application. Oracle Clusterware 11g Release 2 introduces the concept of a server pool. If you only want to start Oracle GoldenGate on a subset of the servers in your cluster then you use a server pool.

For more information about the crsctl add resource command and its options, please refer to the Oracle Clusterware documentation.

If your Oracle GoldenGate software owner (e.g. `mvandewiel`) is not the same as the Oracle Grid infrastructure software owner, then you must set the ownership of the application to the Oracle GoldenGate software owner. Run this command as `root`.

```
GRID_HOME/bin/crsctl setperm resource ggateapp -o mvandewiel
```

Step 4: Start the application

From now on you should always use Oracle Clusterware to start Oracle GoldenGate. Login as `oracle` and execute:

```
GRID_HOME/bin/crsctl start resource ggateapp
```

To check the status of the application:

```
GRID_HOME/bin/crsctl status resource ggateapp
```

For example:

```
[oracle@coe-02 grid]$ crsctl status resource ggateapp
NAME=ggateapp
TYPE=cluster_resource
TARGET=ONLINE
STATE=ONLINE on coe-02

[oracle@coe-02 grid]$
```

Manage the application

When Oracle GoldenGate is running, and you want to move Oracle GoldenGate to run on a different server, you can use the `GRID_HOME/bin/crsctl relocate resource` command with the force option to move the VIP as well (as `oracle`, on any node):

```
[oracle@coe-02 grid]$ crsctl relocate resource ggateapp -f
CRS-2673: Attempting to stop 'ggateapp' on 'coe-01'
CRS-2677: Stop of 'ggateapp' on 'coe-01' succeeded
CRS-2673: Attempting to stop 'mvggatevip' on 'coe-01'
CRS-2677: Stop of 'mvggatevip' on 'coe-01' succeeded
CRS-2672: Attempting to start 'mvggatevip' on 'coe-02'
CRS-2676: Start of 'mvggatevip' on 'coe-02' succeeded
CRS-2672: Attempting to start 'ggateapp' on 'coe-02'
CRS-2676: Start of 'ggateapp' on 'coe-02' succeeded
[oracle@coe-02 grid]$
```

Application relocation is exactly what happens when the server running Oracle GoldenGate crashes.

To stop Oracle GoldenGate, use (as `oracle`):

```
GRID_HOME/bin/crsctl stop resource ggateapp
```

Cleanup

If you want to stop Oracle Clusterware from managing Oracle GoldenGate, and you want to cleanup the changes you made, then:

Stop Oracle GoldenGate (login as `oracle`):

```
GRID_HOME/bin/crsctl stop resource ggateapp
```

Stop the VIP (as `oracle`):

```
GRID_HOME/bin/crsctl stop resource mvggatevip
```

Delete the application `ggateapp` as the application owner (`mvandewiel`) or `root`:

```
GRID_HOME/bin/crsctl delete resource ggateapp
```

Delete the VIP (login as `root`):

```
GRID_HOME/bin/appvipcfg delete -vipname=mvggatevip
```

Delete the agent action script `11gr2_gg_action.scr` at the OS level.

Notes on using Oracle Automatic Storage Management (ASM) 11g Release 2

Oracle GoldenGate supports data capture from an Oracle Database using ASM. This section applies to Oracle ASM 11g Release 2.

A few additional setup steps are required in order to use ASM for Oracle redo and/or archive logs:

- Extract requires a connection into an ASM instance to be able to read the transaction logs. The connection has to go through the Oracle Database listener and because the ASM instance is only mounted (not open) an entry for the ASM instance must be added to the listener configuration file in order to let incoming connections go through. See the Oracle GoldenGate for Windows and Unix Administrator Guide as well as Oracle Support note 340277.1 for more details.
- The tnsnames.ora file(s) or LDAP directory must include entries to the ASM instance(s) in order for connect strings to resolve connection requests.
- The Oracle GoldenGate extract parameter file must include the following line:
- `TRANLOGOPTIONS ASMUSER <user>@<asm>, ASMPASSWORD <password>, ENCRYPTKEY <key>`

An ASM instance is not a regular database instance and does not support the concept of regular database users. As a result the user Oracle GoldenGate uses to connect to ASM is always an administrative user (SYSASM) which would enable startup and shutdown of the instance. The connection will not fail if the instance is down, but Oracle GoldenGate extract will not be able to start because any queries issued against the ASM instance will fail.

If you use Oracle GoldenGate to extract from an Oracle RAC database that uses ASM to store its logs, and your connection to ASM could be routed to any ASM instance in the cluster, then you may run into the situation that extract connects to an ASM instance that happens to be down. In that case extract will crash, although the database may still be running fine on other servers. This will cause the extract process to fall behind.

To prevent this scenario with Oracle ASM 11g Release 2 follow the following steps:

1. Use another dependency to a local resource `ora.asm`. This resource is available if the ASM instance is running. This introduces a slight change to the `crsctl add resource` command (changes highlighted):

   ```
   GRID_HOME/bin/crsctl add resource ggateapp \
        -type cluster_resource \
        -attr
   "ACTION_SCRIPT=/mnt/acfs/oracle/grid/11gr2_gg_action.scr,
   CHECK_INTERVAL=30, START_DEPENDENCIES='hard(mvggatevip,ora.asm)
   pullup(mvggatevip)', STOP_DEPENDENCIES='hard(mvggatevip)'"
   ```

   All other steps to configure Oracle GoldenGate with Oracle Clusterware remain the same, and all configurations discussed in the section Oracle GoldenGate with Oracle Clusterware will work.

2. In the extract parameter file you can include the following TRANLOGOPTIONS parameter (example):

   ```
   TRANLOGOPTIONS ASMUSER sys@asm, ASMPASSWORD asawer14, ENCRYPTKEY DEFAULT
   ```

3. Finally make sure that the connect string `@asm` always connects to the local ASM instance. I.e. the ASM entry in the `tnsnames.ora` specifies a different connection on different nodes. For example on node 1:

   ```
   ASM =
     (DESCRIPTION =
       (ADDRESS_LIST =
         (ADDRESS = (PROTOCOL = TCP)(HOST = coe-01)(PORT = 1523))
       )
       (CONNECT_DATA =
         (SERVICE_NAME = asm)
         (INSTANCE_NAME = +ASM1)
   ```

```
    )
  )
```

On node 2:

```
ASM =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = coe-02)(PORT = 1523))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = asm)
      (INSTANCE_NAME = +ASM2)
    )
  )
```

Extend for subsequent nodes.

Conclusion

Oracle Clusterware is a full functional cluster management solution. Starting with Oracle Database 10g Oracle mandates its users to use Oracle's Clusterware for Oracle RAC cluster management. Starting with Oracle Database 10g Release 2 there are commands to register and manage other applications through Oracle Clusterware.

This best practice document discusses how to address Oracle GoldenGate failover during server failures by using a step-by-step example with sample code to manage Oracle GoldenGate through Oracle Clusterware. Oracle GoldenGate high availability through Oracle Clusterware will help keep Oracle GoldenGate available through operating system and hardware crashes as long as the database is accessible from a surviving node. It should be noted however that there may still be other reasons for Oracle GoldenGate processes to become unavailable and you should continue to use notification procedures to ensure ultimate high availability for Oracle GoldenGate.

Appendix 1: Action Script

Following is a sample agent. This agent will also mount/unmount a DBFS file system upon startup/failover. If you do not use DBFS for recovery-related files then you can remove the calls to the DBFS mount and unmount functions.

```
#!/bin/sh

#goldengate_action.scr

. ~oracle/.profile

[ -z "$1" ]&& echo "ERROR!! Usage $0 <start|stop|abort|clean>"&& exit 99

GGS_HOME=<set the path here>


#specify delay after start before checking for successful start

start_delay_secs=5


#Include the Oracle GoldenGate home in the library path to start GGSCI

export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:${GGS_HOME}


#set the oracle home to the database to ensure Oracle GoldenGate will get

#the right environment settings to be able to connect to the database

export ORACLE_HOME=<set the ORACLE_HOME path here>

export CRS_HOME=<set the CRS_HOME path here>

#Set NLS_LANG otherwise it will default to US7ASCII

export NLS_LANG=American_America.US7ASCII

logfile=/tmp/crs_gg_start.log

\rm ${logfile}
```

```
############################
function log
############################
{
        DATETIME=`date +%d/%m/%y-%H:%M:%S`
         echo $DATETIME "goldengate_action.scr>>" $1
         echo $DATETIME "goldengate_action.scr>>" $1 >> $logfile
}


#check_process validates that a manager process is running at the PID
#that Oracle GoldenGate specifies.
check_process () {

        if ( [ -f "${GGS_HOME}/dirpcs/MGR.pcm" ] )
        then
                pid=`cut -f8 "${GGS_HOME}/dirpcs/MGR.pcm"`

                if [ ${pid} = `ps -e |grep ${pid} |grep mgr |awk '{ print $1 }'` ]
                then
                        #manager process is running on the PID . exit success
                        echo "manager process is running on the PID . exit
success">> /tmp/check.out
                        exit 0
                else
                        #manager process is not running on the PID
                        echo "manager process is not running on the PID" >>
/tmp/check.out
                        exit 1
```

```
                fi


        else
                #manager is not running because there is no PID file
                echo "manager is not running because there is no PID file" >>
/tmp/check.out
                exit 1
        fi
}

#call_ggsci is a generic routine that executes a ggsci command
call_ggsci () {
log "entering call_ggsci"
ggsci_command=$1
#log "about to execute $ggsci_command"
log "id= $USER"
cd ${GGS_HOME}
ggsci_output=`${GGS_HOME}/ggsci << EOF
${ggsci_command}
exit
EOF`
log "got output of : $ggsci_output"
}

#unmount_dbfs will unmount the DBFS file system

unmount_dbfs () {
if ( [ -d ${DBFS_FILE_SYSTEM} ] )
then
```

```
fusermount -u ${DBFS_MOUNT_POINT}
fi
}
```

```
case $1 in
'start')
```

```
 #Updated by Sourav (02/10/201
 # During failover if the "mgr.pcm" file is not deleted at the node crash
 # then Oracle clusterware won't start the manager on the new node assuming the
 # manager process is still running on the failed node. To get around this issue
 # we will delete the "mgr.prm" file before starting up the manager on the new
 # node. We will also delete the other process files with pc* extension and to
 # avoid any file locking issue we will first backup the checkpoint files and then
 # delete them from the dirchk directory.After that we will restore the checkpoint
 # files from  backup to the original location (dirchk directory).
```

```
     log "removing *.pc* files from dirpcs directory..."
        rm -f $GGS_HOME/dirpcs/*.pc*

     log "creating tmp directory to backup checkpoint file...."
         mkdir $GGS_HOME/dirchk/tmp

     log "backing up checkpoint files..."
```

```
          cp $GGS_HOME/dirchk/*.cp* $GGS_HOME/dirchk/tmp



     log "Deleting checkpoint files under dirchk......"
          rm -f $GGS_HOME/dirchk/*.cp*

     log "Restore checkpoint files from backup to dirchk directory...."
          cp $GGS_HOME/dirchk/tmp/*.cp* $GGS_HOME/dirchk

     log "Deleting tmp directory...."
          rm -r $GGS_HOME/dirchk/tmp


        log "starting manager"
        call_ggsci 'start manager'
        #there is a small delay between issuing the start manager command
        #and the process being spawned on the OS . wait before checking
        log "sleeping for start_delay_secs"
        sleep ${start_delay_secs}
        #check whether manager is running and exit accordingly
        check_process
;;
'stop')
        #attempt a clean stop for all non-manager processes
        call_ggsci 'stop er *'
        #ensure everything is stopped
        call_ggsci 'stop er *!'
        #stop manager without (y/n) confirmation
        call_ggsci 'stop manager!'
        #exit success
```

```
        exit 0
;;
'check')
        check_process
        exit 0
;;
'clean')
       #attempt a clean stop for all non-manager processes
       call_ggsci 'stop er *'
       #ensure everything is stopped
       call_ggsci 'stop er *!'
       #in case there are lingering processes
       call_ggsci 'kill er *'
       #stop manager without (y/n) confirmation
       call_ggsci 'stop manager!'
       #unmount DBFS
       unmount_dbfs
       #exit success
       exit 0
;;
'abort')
      #ensure everything is stopped
      call_ggsci 'stop er *!'
      #in case there are lingering processes
      call_ggsci 'kill er *'
      #stop manager without (y/n) confirmation
      call_ggsci 'stop manager!'
      #unmount DBFS
      unmount_dbfs
```

```
      #exit success

      exit 0
;;



esac
```

Appendix 2: Extended Action Script for Oracle ASM

Below is a modified action script that uses an extra routine to update an Oracle GoldenGate library specifying the connection into Oracle ASM. This script may be used for extract out of Oracle ASM 10g Release 2 or Oracle ASM 11g Release 1.

```
#!/bin/sh

#goldengate_action.scr

. ~oracle/.profile

[ -z "$1" ]&& echo "ERROR!! Usage $0 <start|stop|abort|clean>"&& exit 99

GGS_HOME=<set the path here>


#specify delay after start before checking for successful start

start_delay_secs=5


#Include the Oracle GoldenGate home in the library path to start GGSCI

export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:${GGS_HOME}


#set the oracle home to the database to ensure Oracle GoldenGate will get

#the right environment settings to be able to connect to the database

export ORACLE_HOME=<set the ORACLE_HOME path here>

export CRS_HOME=<set the CRS_HOME path here>

#Set NLS_LANG otherwise it will default to US7ASCII

export NLS_LANG=American_America.US7ASCII
```

```
logfile=/tmp/crs_gg_start.log
\rm ${logfile}

#### following section is required to ensure successful connection to ASM ####
set_asm_connection () {

  #get all ASM instances managed by Clusterware
  #output is "name type target state host"
  ASMINSTANCES=`${CRS_HOME}/bin/crs_stat -t |grep asm`

  #define some processing variables
  COUNTER=0

  NEXTONLINE=4
  NEXTNODE=false

  CONTINUE=true

  #specify connection details for the ASM instance
  ASMUSER=sys
  ASMPASSWORD=password
  ASMENCRYPTKEY=key

  #use a separate macro file to connect ASM
  #this file will be overwritten by this process each time it runs
  ASMCONNECTMACRO=${GGS_HOME}/dirprm/asmconn.lib

  #loop through all ASM instance data fields
  #output is "name type target state host"
  #find an ASM instance with state ONLINE
  for b in ${ASMINSTANCES}
  do

    #control whether to continue processing
    if [ ${CONTINUE} = "true" ]
    then

      #node (host) is found after an online status
      if [ ${NEXTNODE} = "true" ]
      then
```

```
        #connection to asm requires retrieval of full crs_stat detail
        #tnsnames for the Oracle GoldenGate environment must contain all
        #individual connections to ASM, pointing to the specfic instances
        #e.g. ASM1, ASM2, ASM3, ASM4
        ASMCONNECT=`${CRS_HOME}/bin/crs_stat |grep asm |grep ${b} |cut -d '.' -f3`

        #output full Oracle GoldenGate ASM connect info to the lib file
        echo "TRANLOGOPTIONS ASMUSER ${ASMUSER}@${ASMCONNECT}, ASMPASSWORD
${ASMPASSWORD}, ENCRYPTKEY ${ASMENCRYPTKEY}" > ${ASMCONNECTMACRO}

        #stop processing
        NEXTNODE="false"
        CONTINUE="false"

      fi

      COUNTER=`expr ${COUNTER} + 1`

      #discard records until the state field is reached
      if [ ${COUNTER} = ${NEXTONLINE} ]
      then

        #check whether state is ONLINE (if not, discard and continue)
        if [ ${b} = "ONLINE" ]
        then

          NEXTNODE=true

        fi

        #there are 5 fields per instance - next state field is 5 ahead
        NEXTONLINE=`expr ${NEXTONLINE} + 5`

      fi

    fi

  done

}
```

```
############################
function log
############################
{
        DATETIME=`date +%d/%m/%y-%H:%M:%S`
         echo $DATETIME "goldengate_action.scr>>" $1
         echo $DATETIME "goldengate_action.scr>>" $1 >> $logfile
}


#check_process validates that a manager process is running at the PID
#that Oracle GoldenGate specifies.
check_process () {

        #reset the ASM connect information in case extracts have to restart
        set_asm_connection

        if ( [ -f "${GGS_HOME}/dirpcs/MGR.pcm" ] )
        then
                pid=`cut -f8 "${GGS_HOME}/dirpcs/MGR.pcm"`

                if [ ${pid} = `ps -e |grep ${pid} |grep mgr |awk '{ print $1 }'` ]
                then
                        #manager process is running on the PID . exit success
                        echo "manager process is running on the PID . exit
success">> /tmp/check.out
                        exit 0
                else
```

```
                        #manager process is not running on the PID
                        echo "manager process is not running on the PID" >>
/tmp/check.out
                        exit 1
                fi

        else
                #manager is not running because there is no PID file
                echo "manager is not running because there is no PID file" >>
/tmp/check.out
                exit 1
        fi
}

#call_ggsci is a generic routine that executes a ggsci command
call_ggsci () {
log "entering call_ggsci"
ggsci_command=$1
#log "about to execute $ggsci_command"
log "id= $USER"
cd ${GGS_HOME}
ggsci_output=`${GGS_HOME}/ggsci << EOF
${ggsci_command}
exit
EOF`
log "got output of : $ggsci_output"
}

#unmount_dbfs will unmount the DBFS file system
```

```
unmount_dbfs () {
if ( [ -d ${DBFS_FILE_SYSTEM} ] )
then
fusermount -u ${DBFS_MOUNT_POINT}
fi
}



case $1 in
'start')

        log "setting up asm connect string"
        #make sure to initialize an ASM connect string
        set_asm_connection

#Updated by Sourav (02/10/201
 # During failover if the "mgr.pcm" file is not deleted at the node crash
 # then Oracle clusterware won't start the manager on the new node assuming the
 # manager process is still running on the failed node. To get around this issue
 # we will delete the "mgr.prm" file before starting up the manager on the new
 # node. We will also delete the other process files with pc* extension and to
 # avoid any file locking issue we will first backup the checkpoint files and then
 # delete them from the dirchk directory.After that we will restore the checkpoint
 # files from backup to the original location (dirchk directory).
```

```
log "removing *.pc* files from dirpcs directory..."
   rm -f $GGS_HOME/dirpcs/*.pc*

log "creating tmp directory to backup checkpoint file...."
    mkdir $GGS_HOME/dirchk/tmp

log "backing up checkpoint files..."
    cp $GGS_HOME/dirchk/*.cp* $GGS_HOME/dirchk/tmp


log "Deleting checkpoint files under dirchk......"
     rm -f $GGS_HOME/dirchk/*.cp*

log "Restore checkpoint files from backup to dirchk directory...."
     cp $GGS_HOME/dirchk/tmp/*.cp* $GGS_HOME/dirchk

log "Deleting tmp directory...."
     rm -r $GGS_HOME/dirchk/tmp


  log "starting manager"
  call_ggsci 'start manager'
  #there is a small delay between issuing the start manager command
  #and the process being spawned on the OS . wait before checking
  log "sleeping for start_delay_secs"
  sleep ${start_delay_secs}
  #check whether manager is running and exit accordingly
  check_process
```

```
;;
'stop')
        #attempt a clean stop for all non-manager processes
        call_ggsci 'stop er *'
        #ensure everything is stopped
        call_ggsci 'stop er *!'
        #stop manager without (y/n) confirmation
        call_ggsci 'stop manager!'
        #exit success
        exit 0
;;
'check')
        check_process
        exit 0
;;
'clean')
       #attempt a clean stop for all non-manager processes
       call_ggsci 'stop er *'
       #ensure everything is stopped
       call_ggsci 'stop er *!'
       #in case there are lingering processes
       call_gsci 'kill er *'
       #stop manager without (y/n) confirmation
       call_ggsci 'stop manager!'
       #unmount DBFS
       unmount_dbfs
       #exit success
       exit 0
;;
```

```
'abort')
      #ensure everything is stopped
      call_ggsci 'stop er *!'
      #in case there are lingering processes
      call_gsci 'kill er *'
      #stop manager without (y/n) confirmation
      call_ggsci 'stop manager!'
      #unmount DBFS
      unmount_dbfs
      #exit success
      exit 0
;;


esac


#Updated by Sourav (11/12/2010)
```

Appendix 3:

Workaround for BUG:9964188 - APPVIPCFG PRODUCES ERROR CRS-0160

Applies to:

Oracle Server - Enterprise Edition - Version: 11.2.0.1.0 to 11.2.0.1.0 - Release: 11.2 to 11.2
Information in this document applies to any platform.

Symptoms

Invoking the appvipcfg.pl script after PSU2 is applied to the Grid Infrastructure home results in the following error:

CRS-0160: The attribute '"USR_ORA_VIP' is not supported in this resource

Changes

When the following command is run:

root# /opt/grid/11.2.0/bin/appvipcfg create -network=1 -ip=10.56.200.26
-vipname=testfororacle -user=grid

The output (including the error) produced is:

Production Copyright 2007, 2008, Oracle.All rights reserved
2010-08-02 13:23:36: Skipping type creation
2010-08-02 13:23:36: Create the Resource
2010-08-02 13:23:36: Executing cmd: /opt/grid/11.2.0/bin/crsctl add resource
testfororacle -type app.appvip.type -attr
root:rwx,pgrp:root:r-x,other::r--,user:grid:r-x'"
CRS-0160: The attribute '"USR_ORA_VIP' is not supported in this resource type.
CRS-4000: Command Add failed, or completed with errors.
Command return code of 1 (256) from command: /opt/grid/11.2.0/bin/crsctl add
resource testfororacle -type app.appvip.type -attr
root:rwx,pgrp:root:r-x,other::r--,user:grid:r-x'"
2010-08-02 13:23:36: ###### Begin Error Stack Trace ######
2010-08-02 13:23:36: Package File Line Calling
2010-08-02 13:23:36: --------------- -------------------- ----
----
2010-08-02 13:23:36: 1: crsconfig_lib crsconfig_lib.pm 7496
crsconfig_lib::error
2010-08-02 13:23:36: 2: main appvipcfg.pl 216
crsconfig_lib::system_cmd
2010-08-02 13:23:36: 3: main appvipcfg.pl 95
main::config

2010-08-02 13:23:36: 4: main appvipcfg.pl 76
main::process_arguments
2010-08-02 13:23:36: ######## End Error Stack Trace ########

Cause

Through the investigation into code defect Bug 9964188, it was confirmed that this problem is due to extra characters (namely: \") that were included inadvertently in the perl script: "appvipcfg.pl"

Solution

The fix for Bug 9964188 will be included in a future PSU patch.

For now, the following workaround may be implemented:

1. Make a backup copy of the file "appvipcfg.pl"

2. Modify the appvipcfg.pl script to remove the extra \" as follows:

These are the two lines we need to modify **BEFORE** modification:

```
211 "\"USR_ORA_VIP=$ipaddress",
...
214 "ACL=$acllist\""
```

These are the two lines we need to modify **AFTER** modification:

```
211 "USR_ORA_VIP=$ipaddress",
...
214 "ACL=$acllist"
```

Workaround for BUG :9767810 –CRS-2518 while registering Goldengate as a resource in Oracle 11.2 clusterware.

Solution: Check support doc id 948456.1

White Paper Title: Oracle GoldenGate high availability with Oracle Clusterware
March 2010
Author: Mark Van de Wiel
Contributing Authors:
Sourav Bhattacharya

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com

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

0109