# Oracle GoldenGate and RAC High Availability

## Introduction

This resource will start the GoldenGate manager process only.

Use the AUTOSTART parameter within the manager parameter file in order to start the EXTRACT and REPLICAT processes.

The resource can also fail over to the surviving node if one node crashes., although this will only work if GoldenGate is on shared storage.

If you stop the manager process from the command line, CRS will automatically attempt to restart it.

Unless of course if the resource has a target of OFFLINE.

These instructions are only applicable for Oracle 10gR2.

In 12cR1 there is a bundled agent for GoldenGate High Availability.

For full instructions see Document 1313703.1 on My Oracle Support

## Create Application VIP

As oracle user …  
crs\_profile -create ggatevip -t application -a /u01/app/crs/10.2.0/crs\_1/bin/usrvip -o oi=nxge2,ov=10.251.199.163,on=255.255.252.0  
crs\_register ggatevip

oi = Public Interface  
ov = New VIP address  
on = Subnet mask of the public IP address

As root user …  
./crs\_setperm ggatevip -o root  
./crs\_setperm ggatevip -u user:oracle:r-x

As oracle user …  
crs\_start ggatevip  
crs\_stat ggatevip -t

## Create Action Script

Create a file called goldengate\_action.scr using the script at the end of this document.

Copy file to /u01/app/crs/10.2.0/crs\_1/crs/public on all RAC nodes and make sure it is executable.

Within the script check the environment variables GGS\_HOME and ORACLE\_SID to make sure they are correct.

## Create Application Resource

crs\_profile -create goldengate\_app -t application -r ggatevip -a /u01/app/crs/10.2.0/crs\_1/crs/public/goldengate\_action.scr -o ci=10

crs\_register goldengate\_app

crs\_start goldengate\_app

crs\_stat goldengate\_app –t

ci = Check Interval in seconds

molintdb1:molintbs1-->crs\_stat –t | grep gate

Name Type Target State Host

------------------------------------------------------------

ggatevip application ONLINE ONLINE molintdb1

goldengate\_app application ONLINE ONLINE molintdb1

## Action Script

#!/bin/sh

#goldengate\_action.scr

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

GGS\_HOME=/u01/app/oracle/product/11.2.0/goldengate

export GGS\_HOME

#specify delay after start before checking for successful start

start\_delay\_secs=5

ORACLE\_HOME=/u01/app/oracle/product/10.2.0/db\_1

CRS\_HOME=/u01/app/crs/10.2.0/crs\_1

LD\_LIBRARY\_PATH=$ORACLE\_HOME/lib:$GGS\_HOME

ORACLE\_SID=molintbs1

export LD\_LIBRARY\_PATH ORACLE\_HOME CRS\_HOME ORACLE\_SID

LD\_LIBRARY\_PATH\_64=$ORACLE\_HOME/lib:$GGS\_HOME

export LD\_LIBRARY\_PATH\_64

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

}

# Executes a ggsci command

call\_ggsci () {

ggsci\_command=$1

cd ${GGS\_HOME}

ggsci\_output=`${GGS\_HOME}/ggsci << EOF

${ggsci\_command}

exit

EOF`

}

case $1 in

'start')

rm -f $GGS\_HOME/dirpcs/\*.pc\*

mkdir $GGS\_HOME/dirchk/tmp

cp $GGS\_HOME/dirchk/\*.cp\* $GGS\_HOME/dirchk/tmp

rm -f $GGS\_HOME/dirchk/\*.cp\*

cp $GGS\_HOME/dirchk/tmp/\*.cp\* $GGS\_HOME/dirchk

rm -r $GGS\_HOME/dirchk/tmp

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

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!'

#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!'

#exit success

exit 0

;;

esac