	Test Plan for Oracle Database & Grid Infrastructure 12c	Release 1 - 12.1.0.2.0	
#	**Test Description/Procedure**	**Expected Results/Notes**	**P/F**
1	**Check & Test Clusterware.**		

A	**Check: Log & Trace Files.**
	Clusterware alert log:
	`$ADR_HOME/crs/<hostname>/crs/trace/alert.log`
	Listener log files:
	`$ADR_HOME/tnslsnr/<hostname>/<listener>/trace/listener.log`
	CRS process trace files:
	`$ADR_HOME/crs/<hostname>/crs/trace/*`
		Clusterware ADR_HOME is: `/u01/app/grid/diag/`
		Check for errors/warnings in these files.	

B	**Check: Resource TARGET & STATE columns.**
	As grid user, run:
	`crs_stat -t`
	`crsctl status resource -t`
		Check for TARGET ONLINE and STATE OFFLINE combinations.	

C	**Check: Cluster Verification.**
	As grid user, run:
	`crsctl check cluster -all`
		All nodes should return:
		`CRS-4537: Cluster Ready Services is online`
		`CRS-4529: Cluster Synchronization Services is online`
		`CRS-4533: Event Manager is online`	

D	**Check: CRS Verification.**
	As grid user, run on each node:
	`crsctl check crs`
		All nodes should return:
		`CRS-4638: Oracle High Availability Services is online`
		`CRS-4537: Cluster Ready Services is online`
		`CRS-4529: Cluster Synchronization Services is online`
		`CRS-4533: Event Manager is online`	

E	**Check: CTSS.**
	As grid user, run on each node:
	`crsctl check ctss`
		All nodes should return:
		`CRS-4701: The Cluster Time Synchronization Service is in Active mode.`
		`CRS-4702: Offset (in msec): 0`	

F	**Check: DNS.**
	As grid user, run on each node:
	`crsctl query dns -servers`
		All nodes should return:
		`CRS-10018: the following configuration was found on the system:`
		`CRS-10019: There are 1 domains in search order. They are:`
		`(your-domain-name)`
		`CRS-10022: There are 1 name servers. They are:`
		`(IP-of-your-DNS-server(s))`
		`CRS-10020: number of retry attempts for name lookup is: 4`
		`CRS-10021: timeout for each name lookup is: 5`	

#	Test Description/Procedure	Expected Results/Notes	P/F
1	**Check & Test Clusterware.**		
G	**Check: Votedisk(s).** As grid user, run: `crsctl query css votedisk` Verify locations using OS or ASM commands.	Command should return valid locations of all the Votedisks.	

H	**Check: OCR File(s).**
	As grid user, run:
	`cat /etc/oracle/ocr.loc`
	`ocrcheck -details`
	Verify using OS or ASM commands.
		Running the ocrcheck command as root also checks for OCR logical corruption.
		Commands should return valid locations of all OCR files.	

I	**Check: OCR Backups.** As grid user, run: `ocrconfig -showbackup` Verify backup files using OS commands.	OCR backups are taken every 4 hours. Check for them on any node if the cluster has been up and running long enough. Command should list valid OCR backup files.	

J	**Check: OLR.**
	As grid user, run on each node:
	`cat /etc/oracle/olr.loc`
	`ocrcheck -local -config`
	Verify the OLR file exists using OS commands.
		On each node check the location specified in the olr.loc file agrees with the output returned by the ocrcheck command.	

K	**Check: GPnP.** Check for the existence of the GPnP profile XML file on each node. Default location is: `$GI_HOME/gpnp/<hostname>/profiles/peer/profile.xml`	The profile.xml can view viewed with Firefox. Check the configuration information it contains looks correct.	

#	Test Description/Procedure	Expected Results/Notes	P/F
1	**Check & Test Clusterware.**		

L	**Check: SCAN VIPs.**
	
	As grid user, run:
	`srvctl status scan`
	`srvctl config scan`
	
	Then verify the output at the OS level using:
	`ps -ef | grep SCAN`
		The LISTENER_SCANn listeners are distributed around the cluster nodes and run on the SCAN VIPs. Verify they are running on the nodes and IPs reported by the srvctl commands.
		
		The output from status scan should look like this:
		`SCAN VIP scan1 is enabled`
		`SCAN VIP scan1 is running on node <hostname>`
		`etc.`
		
		The output from config scan should look like this:
		`SCAN name: <SCAN-NAME>.<your-domain-name>, Network: 1`
		`Subnet IPv4: xxx.xxx.0.0/255.255.255.0/eth0, static`
		`Subnet IPv6:`
		`SCAN 0 IPv4 VIP: <SCAN-IP-ADDRESS>`
		`SCAN VIP is enabled.`
		`SCAN VIP is individually enabled on nodes:`
		`SCAN VIP is individually disabled on nodes:`
		`etc.`	

M	**Check: Node VIPs.**
	
	As grid user, run:
	`srvctl status vip -node <hostname>`
	
	Then verify the status of each VIP by using ping.
		The output should look like this:
		`VIP <hostname>-vip.<your-domain-name> is enabled`
		`VIP <hostname>-vip.<your-domain-name> is running on node: <hostname>`	

N	**Check: Nodeapps.**
	
	As grid user, run:
	`srvctl status nodeapps`
		Nodeapps is made up of the VIPs, the network & the Oracle Notification Service (ONS). The output should look like this:
		`VIP <hostname>-vip.<your-domain-name> is enabled`
		`VIP <hostname>-vip.<your-domain-name> is running on node: <hostname>`
		(repeated for each VIP)
		`Network is enabled`
		`Network is running on node: <hostname>`
		(repeated for each node)
		`ONS is enabled`
		`ONS daemon is running on node: <hostname>`
		(repeated for each node)	

O	**Check: Node Participation.** As grid user, run: `olsnodes -n -i -s -t -a`	-n : print node name and node number -i : VIP IP address or VIP name -s : node status (active| inactive) -t : node type (pinned | unpinned) -a : node role (hub | leaf) Verify the output reports the correct information.	

#	Test Description/Procedure	Expected Results/Notes	P/F
1	**Check & Test Clusterware.**		

P	**Test: Unplanned SCAN Listener Termination.**
	
	As the root user, pick any SCAN Listener process id:
	`ps -ef | grep SCAN`
	
	Kill the process id of your selected SCAN Listener:
	`kill -9 <process_id>`
		New connections via SQL*Plus should succeed and the SCAN Listener process should be re-started automatically.
		Note the process id of the new SCAN Listener process, then verify the re-start of that process in the SCAN listener alert log.	

Q	**Test: CRSD Process Termination.**
	
	As the root user, locate the crsd process:
	`ps -ef | grep crsd.bin | grep -v grep`
	
	Kill the process id of crsd.bin
	`kill -9 <process_id>`
		The crsd.bin process should be re-started automatically.
		Note the process id of the new crsd.bin process, then verify the re-start of that process id in the Clusterware alert log.	

R	**Test: EVMD Process Termination.**
	
	As the root user, locate the crsd process:
	`ps -ef | grep evmd.bin | grep -v grep`
	
	Kill the process id of evmd.bin
	`kill -9 <process_id>`
		The evmd.bin process should be re-started automatically.
		Note the process id of the new crsd.bin process, then verify the re-start of that process id in the Clusterware alert log.	

S	**Test: CSSD Process Termination.**
	
	As the root user, locate the crsd process:
	`ps -ef | grep ocssd.bin | grep -v grep`
	
	Kill the process id of ocssd.bin
	`kill -9 <process_id>`
		The node will reboot itself. Its cluster resources should migrate to the surviving node(s). After the rebooted node comes back online, cluster resources should migrate back to it. The ora.cvu resource will stay OFFLINE/OFFLINE.	

#	Test Description/Procedure	Expected Results/Notes	P/F
2	**Check & Test ASM.**		
A	**Check: Log & Trace Files.** ASM alert log: `$ADR_HOME/asm/+asm/<ASM_instance>/trace/alert_<ASM_instance>.log`	Clusterware ADR_HOME is: `/u01/app/grid/diag`. Check for errors/warnings in these files.	

B	**Check: ASM Instances.**
	As grid user, run:
	`srvctl status asm`
	Verify by logging in to ASM instance on each node using SQL*Plus or ASMCMD:
	`sqlplus / as sysasm`
	`asmcmd`
		The output should look like this:
		`ASM is running on <hostname1>,<hostname2>,…,<hostnameN>`
		Logins to the ASM instance should be successful with no errors.	

C	**Check: ASM Diskgroups.**
	As grid user, run:
	`echo "select name from v\$asm_diskgroup;" | sqlplus -s / as sysasm`
	Then for each ASM Diskgroup name returned, run:
	`for dg in <DG_NAME1> <DG_NAME2> <DG_NAMEn>`
	`> do`
	`> srvctl status diskgroup -diskgroup ${dg} -detail`
	`> done`
		The output should look like this:
		`Disk Group <DG_NAME1> is running on <hostname1>,<hostname2>,...`
		`Disk Group <DG_NAME1> is enabled`
		(repeats for each DG_NAME)	

D	**Check: ASM Diskgroup Metadata.**
	Log into the ASM instance using SQL*Plus then run:
	`ALTER DISKGROUP <DG_NAME> CHECK ALL;`
	or
	Log into ASMCMD then run:
	`chkdg <DB_NAME>`
		If there are no errors SQL*Plus and ASMCMD will return:
		`Diskgroup altered.`	

<table>
<tr><th>#</th><th>Test Description/Procedure</th><th>Expected Results/Notes</th><th>P/F</th></tr>
<tr><td><b>2</b></td><td><b>Check &amp; Test ASM.</b></td><td></td><td></td></tr>
<tr><td>E</td><td>

Check: ASM Disks.

As grid user, run:
In SQL*Plus:

```
select dg.name DG,
       d.name Disk,
       decode(d.GROUP_NUMBER,
              0,'Unallocated',
              'In Use') State,
       d.path
from   v$asm_disk d
       left outer join
       v$asm_diskgroup dg
on     dg.group_number = d.group_number
order by
       dg.name,
       d.path;
```

or
In ASMCMD:

```
lsdsk --discovery
```

</td><td>Verify all configured ASM Disks are visible via the ASM instance or ASMCMD.</td><td></td></tr>
<tr><td>F</td><td>

Check: ASM Clients.

As grid user, on each node run:

```
select dg.NAME,
       c.INSTANCE_NAME,
       c.DB_NAME,
       c.CLUSTER_NAME,
       c.STATUS
from   v$asm_client c,
       v$asm_diskgroup dg
where  c.GROUP_NUMBER = dg.GROUP_NUMBER
order by
       c.DB_NAME;
```

</td><td>All nodes apart from one should list the names of the ASM Diskgroups the local node's database instance(s) are using, along with a row for DB_NAME +ASM which will be using the ASM Diskgroup where the Votedisk(s) and OCR file(s) are located. All these rows should show a STATUS of CONNECTED.

The one exception will be the node where the GIMR database is running. An additional database with DB_NAME _mgmtdb and instance name -MGMTDB should be displayed also with a STATUS of CONNECTED.</td><td></td></tr>
<tr><td>G</td><td>

Test: Unplanned ASM Instance Termination.

As the root user, locate the ASM instance pmon process:

```
ps -ef | grep pmon | grep -v grep
```

Kill the process id of ocssd.bin

```
kill -9 <process_id>
```

</td><td>This should abort both the ASM instance and any connected RDBMS instance clients. The ASM instance failure should be detected and re-started, followed by the RDBMS instance(s) re-starting.</td><td></td></tr>
</table>

#	Test Description/Procedure	Expected Results/Notes	P/F
2	**Check & Test ASM.**		

H	**Test: Add an ASM Disk to an ASM Diskgroup.**
	
	As grid user:
	Locate an unallocated ASM Disk:
	
	`select dg.name,`
	`       d.name,`
	`       decode(d.GROUP_NUMBER,0,'Unallocated'),`
	`       d.path`
	`from   v$asm_disk d`
	`       left outer join`
	`       v$asm_diskgroup dg`
	`on     dg.group_number = d.group_number`
	`order by`
	`       d.path;`
	
	Add an unallocated ASM Disk to an existing ASM Diskgroup:
	
	`alter diskgroup <DG_NAME> add disk '<ASM_DISK_PATH>’;`
	
	In another session, query `V$ASM_OPERATION`:
	
	`select GROUP_NUMBER,`
	`       OPERATION,`
	`       POWER,`
	`       EST_WORK,`
	`       EST_RATE,`
	`       EST_MINUTES`
	`from   v$asm_operation;`
		This should cause an ASM Diskgroup rebalance operation. If the rebalance is taking too long, the ASM Diskgroup power setting can be changed to a higher value (up to 11):
		
		`alter diskgroup <DG_NAME> rebalance power <NEW_POWER_LEVEL>;`
		
		The `alter diskgroup…add disk` command should return to the SQL prompt after a few moments displaying this message:
		
		`Diskgroup altered.`
		
		The rebalance operation could continue for some time depending upon how full the existing ASM Diskgroup’s disks were. When the rebalance operation completes, the associated rows in `V$ASM_OPERATION` will be deleted automatically.	

#	Test Description/Procedure	Expected Results/Notes	P/F
2	**Check & Test ASM.**		

I	**Test: Drop an ASM Disk from an ASM Diskgroup.**
	
	As grid user:
	Locate the name of the ASM Disk you wish to drop:
	
	`select dg.name,`
	`       d.name,`
	`       decode(d.GROUP_NUMBER,0,'Unallocated'),`
	`       d.path`
	`from   v$asm_disk d`
	`       left outer join`
	`       v$asm_diskgroup dg`
	`on     dg.group_number = d.group_number`
	`order by`
	`       d.path;`
	
	Drop the ASM Disk:
	
	`alter diskgroup <DG_NAME> drop disk <ASM_DISK_NAME>;`
	
	In another session, query V$ASM_OPERATION:
	
	`select GROUP_NUMBER,`
	`       OPERATION,`
	`       POWER,`
	`       EST_WORK,`
	`       EST_RATE,`
	`       EST_MINUTES`
	`from   v$asm_operation;`
		This should cause an ASM Diskgroup rebalance operation. If the rebalance is taking too long, the ASM Diskgroup power setting can be changed to a higher value (up to 11):
		
		`alter diskgroup <DG_NAME> rebalance power <NEW_POWER_LEVEL>;`
		
		The `alter diskgroup…drop disk` command should return to the SQL prompt after a few moments, but the rebalance operation could continue for some time depending upon how much data was on the dropped ASM Disk. When the rebalance operation completes, the associated rows in `V$ASM_OPERATION` will be deleted automatically.	

#	Test Description/Procedure	Expected Results/Notes	P/F
2	**Check & Test ASM.**		

J	**Test: Undrop an ASM Disk.**
	
	As grid user:
	Locate the name of the ASM Disk you wish to drop:
	
	`select dg.name,`
	`       d.name,`
	`       decode(d.GROUP_NUMBER,0,'Unallocated'),`
	`       d.path`
	`from   v$asm_disk d`
	`       left outer join`
	`       v$asm_diskgroup dg`
	`on     dg.group_number = d.group_number`
	`order by`
	`       d.path;`
	
	Drop the ASM Disk:
	
	`alter diskgroup <DG_NAME> drop disk <ASM_DISK_NAME>;`
	
	Undrop the ASM Disk:
	
	`alter diskgroup <DG_NAME> undrop disks;`
		After the `ALTER DISKGROUP…DROP DISK` command, the `STATE` column for that ASM Disk in the `V$ASM_DISK` view should show `DROPPING`.
		
		After the `ALTER DISKGROUP…UNDROP DISKS` command, the `STATE` column for that ASM Disk in the `V$ASM_DISK` view should show `NORMAL`.	

<table>
<tr><th>#</th><th>Test Description/Procedure</th><th>Expected Results/Notes</th><th>P/F</th></tr>
<tr><td><b>2</b></td><td><b>Check &amp; Test ASM.</b></td><td></td><td></td></tr>
<tr><td>K</td><td>

Test: Add an ASM Diskgroup.

As grid user, run:
Locate unallocated ASM Disks:

```
select dg.name,
       d.name,
       decode(d.GROUP_NUMBER,0,'Unallocated'),
       d.path
from   v$asm_disk d
       left outer join
       v$asm_diskgroup dg
on     dg.group_number = d.group_number
order by
       d.path;

create diskgroup <NEW_DG_NAME>
external redundancy
disk '<UNALLOCATED_DISK_PATH>',
     '<UNALLOCATED_DISK_PATH>';
```

Check the new ASM Diskgroup in SQL*Plus:

```
select group_number, name
from   v$asm_diskgroup;
```

or in ASMCMD:

```
lsdg
```

</td><td>

The command should respond with:

```
Diskgroup created.
```

Verify the existence of the ASM Diskgroup using SQL*Plus or ASMCMD.

</td><td></td></tr>
<tr><td>L</td><td>

Test: Drop an ASM Diskgroup.

As grid user, run:

```
drop diskgroup <NEW_DG_NAME>;
```

Check the remaining ASM Diskgroup in SQL*Plus:

```
select group_number, name
from   v$asm_diskgroup;
```

or in ASMCMD:

```
lsdg
```

</td><td>

The command should respond with:

```
Diskgroup dropped.
```

Verify the ASM Diskgroup has been dropped and the ASM Disks returned to the unallocated pool.

</td><td></td></tr>
</table>

#	Test Description/Procedure	Expected Results/Notes	P/F
3	**Check & Test Databases & Instances.**		
A	**Check: GIMR Database Instance Log File.** Instance alert log: `$ADR_HOME/rdbms/_mgmtdb/-MGMTDB/trace/alert_-MGMTDB.log`	The -MGMTDB instance only runs on one node at a time within the cluster. Clusterware ADR_HOME is: `/u01/app/grid/diag`. Check for errors/warnings in these files.	
B	**Check: User Database Instance Log File.** Instance alert log: `$ADR_HOME/rdbms/<db_name>/<inst_name>/trace/alert_<inst_name>.log`	Database ADR_HOME is: `/u01/app/oracle/diag`. Check for errors/warnings in these files.	

C	**Check: User RAC Database & Instance.**
	
	As oracle user, run:
	`srvctl status database -d <DB_NAME>`
		The output should look like this:
		`Instance <INST_NAME1> is running on node <hostname1>`
		`Instance <INST_NAME2> is running on node <hostname2>`
		`Instance <INST_NAME2> is running on node <hostnameN>`	

D	**Check: User RAC Database DBVERIFY.**
	
	As oracle user, find the list of database datafiles:
	`select name from v$datafile;`
	
	For each datafile returned, run:
	`dbv parfile=dbv_parfile.txt`
	
	where dbv_parfile.txt contains:
	`FILE='<path-to-datafile'`
	`LOGFILE=<path-to-logfile>`
	`FEEDBACK=10`
	`USERID=<DBA_USERID>/<DBA_PASSWD>`
		This assumes the database block size is 8K and that the whole file should be verified.
		Examine the logfile for errors.	

E	**Test: Unplanned GIMR Listener Termination.**
	
	As root user:
	Find the process id of the MGMTLSNR process:
	`ps -ef | grep -v grep | grep MGMTLSNR`
	
	Then run:
	`kill -9 <process_id>`
	
	As grid user, verify the running listener:
	`lsnrctl status MGMTLSNR`
		The failure is detected and the listener re-started.
		The lsnrctl status command should return output similar to this:
		`Services Summary...`
		`Service "-MGMTDBXDB" has 1 instance(s).`
		`  Instance "-MGMTDB", status READY, has 1 handler(s) for this service...`
		`Service "_mgmtdb" has 1 instance(s).`
		`  Instance "-MGMTDB", status READY, has 1 handler(s) for this service...`
		`Service "cluster1" has 1 instance(s).`
		`  Instance "-MGMTDB", status READY, has 1 handler(s) for this service...`	

#	Test Description/Procedure	Expected Results/Notes	P/F
3	**Check & Test Databases & Instances.**		

F	**Test: Unplanned GIMR Instance Termination.**
	
	As root user:
	Find the process id of the -MGMTDB pmon process:
	`ps -ef | grep -v grep | grep pmon`
	
	Then run:
	`kill -9 <process_id>`
	
	As grid user, verify the -MGMTDB instance has re-started:
	`ps -ef | grep -v grep | grep pmon`
	
	Check the instance re-registers with the MGMTLSNR:
	`lsnrctl status MGMTLSNR`
		The failure is detected and the instance re-started on the same node.	

G	**Test: Unplanned User Instance Termination.**
	
	As root user:
	Find the process id of a user instance pmon process:
	`ps -ef | grep -v grep | grep pmon`
	
	Then run:
	`kill -9 <process_id>`
	
	Check the cluster re-configurations with:
	`crsctl status resource -t`
		The cluster should re-configure with services moving to surviving instances on other nodes. Any client connections should fail over to surviving instances. A surviving instance should perform instance recovery (rolling back uncommitted transactions for the failed instance). The failed instance should be re-started, followed by a cluster re-configuration with services moving back to the re-started instance.	

#	Test Description/Procedure	Expected Results/Notes	P/F
3	**Check & Test Databases & Instances.**		

H	**Test: Restart User RAC Database.**
	As oracle user, run:
	`srvctl stop database -d <DB_NAME>`
	Verify the instances are down:
	`ps -ef | grep -v grep | grep pmon`
	Re-start the database:
	`srvctl start database -d <DB_NAME>`
	Verify the instances are up:
	`ps -ef | grep -v grep | grep pmon`
	Verify the OCR has been updated:
	As grid user, run:
	`crsctl status resource -t`
		The `srvctl stop` command will shutdown all the RAC database's instances. It waits until the last instance is down, then returns to an OS prompt.
		The `srvctl start` command will re-start all the RAC database's instances. It waits until all the instances have re-started, re-registered with the various listeners and updated the OCR file(s).	

I	**Test: Planned User Instance Shutdown.**
	As oracle user, run:
	`srvctl stop instance -d <DB_NAME> -I <INSTANCE_NAME>`
	Verify the instance is stopped:
	`crsctl status resource -t`
		The srvctl command takes the named instance offline, then shuts it down. All other cluster resources running on the affected node should remain up and running.	

#	Test Description/Procedure	Expected Results/Notes	P/F
3	**Check & Test Databases & Instances.**		

J	**Test: User Session TAF.**
	
	As oracle user, connect to an instance using the
	`<connect-string-name>:`
	`sqlplus <user>/<pwd>@<connect-string-name>`
	`select instance_name from v$instance;`
	
	In another session, run:
	`srvctl stop instance -d <DB_NAME> -I <INSTANCE_NAME>`
	
	In SQL*Plus session, run:
	`select instance_name from v$instance;`
		Configure a client tnsnames.ora file to include an entry for the RAC database:
		
		`<connect-string-name> =`
		`  (DESCRIPTION =`
		`    (ADDRESS = (PROTOCOL = TCP)`
		`               (HOST = <cluster-scan-name>.<domain-name>)`
		`               (PORT = 1521))`
		`    (CONNECT_DATA =`
		`      (SERVER = DEDICATED)`
		`      (SERVICE_NAME = <service-name>)`
		`      (FAILOVER_MODE =`
		`        (TYPE=SESSION)`
		`        (METHOD=BASIC)`
		`        (RETRIES=10)`
		`        (DELAY=10)`
		`      )`
		`    )`
		`  )`
		
		After the instance you originally connected to is shutdown, your session should transition to a surviving instance.	

#	Test Description/Procedure	Expected Results/Notes	P/F
4	**Test System & Cluster.**		

A	**Test: Node Reboot.**
	As root user, run:
	`reboot`
	As grid user, monitor the cluster re-configuration:
	`crsctl status resource -t`
		Reboot all nodes one at a time.
		Review the transition of processes to running nodes.
		The rebooted node’s VIP, SCAN VIP(s), SCAN Listener(s) and Services fail over to another node.
		Ensure rebooted node comes back online and re-starts the correct processes.
		Review the transition of processes back to the rebooted node.	

B	**Test: Node Restart.**
	As root user, run:
	`shutdown -h now`
	As grid user, monitor the cluster re-configuration:
	`crsctl status resource -t`
	Once the cluster re-configuration has completed with the node shutdown, re-start it.
		Review the transition of processes to running nodes.
		The shutdown node’s VIP, SCAN VIP(s), SCAN Listener(s) and Services fail over to another node and stay there. After the shutdown node has re-started, the VIP, SCAN VIP(s), SCAN Listener(s) and Services should transition back to the newly started up node.	

C	**Test: Cluster Restart.**
	As root user, run:
	`cd $GI_HOME/bin`
	`./crsctl stop cluster -all`
	Wait for control to return to the OS prompt, then:
	`./crsctl check cluster -all`
	To re-start the cluster, run:
	`./crsctl start cluster -all`
	To verify all processes and GI resources are back online:
	`./crsctl status resource -t`
		Grid Infrastructure home GI_HOME is `/u01/app/12.1.0/grid`.
		The crsctl stop command should generate dozens on messages beginning with CRS-nnnn with text including:
		`“Attempting to stop ‘<GI resource>’ on <node>”`
		`“Stop of <GI resource> on <node> succeeded”`
		Once control returns to the OS prompt, the check cluster command should return this output for each node in the cluster:
		`<node>:`
		`CRS-4535: Cannot communicate with Cluster Ready Services`
		`CRS-4530: Communications failure contacting Cluster Synchronization Services daemon`
		`CRS-4534: Cannot communicate with Event Manager`
		The output of messages from the `crsctl start` command are fewer in number, but following a similar format to the stop messages. The MGMTDB database comes back online after all the other GI resources have been re-started. This is followed by the RAC database coming back online. Note, GI resources will not necessarily re-start on the same nodes on which they were running before the cluster was stopped.	

#	Test Description/Procedure	Expected Results/Notes	P/F
5	**Health Checks & ORAchk.**		
A	**Check: Cluster Health Check.** As grid user, run: `cluvfy comp healthcheck -collect cluster -bestpractice > cluster_hc.txt`	Review the contents of the output file cluster_hc.txt.	
B	**Check: RAC Database Health Check.** As grid user, run: `cluvfy comp healthcheck -collect database -bestpractice > database_hc.txt`	Review the contents of the output file database_hc.txt.	
C	**Check: Run ORAchk.** Download ORAchk, then as oracle user, unzip it then run: `orachk -a -o verbose`	The ORAchk software can be downloaded via MOS Doc ID: 1268927.2. Examine the output report and take any necessary corrective action(s).	