Cisco WebEx Social Disaster Recovery Using Snapshots, Release 3.0

Rev. 02 (10/31/12)

This procedure describes how to use the VMware snapshotting functionality to effectively create backups of your Cisco WebEx Social nodes.

This is applicable if Oracle has been deployed as either of the following:

- a standard Cisco WebEx Social RDBMS Store node
- a custom single virtual machine in the same ESX infrastructure as the Cisco WebEx Social nodes.

The procedure is *not* applicable when Oracle has been deployed on either of the following:

- one or more physical machines
- a separate ESX infrastructure
- multiple virtual machines.

This document is organized as follows:

How it Works

This backup method utilizes standard ESX functionality to create a snapshot of each virtual machine (VM) at a synchronized time and then build a detached clone of the VM based on the snapshot. “Clone” is a detached virtual machine which can be processed like a separate virtual machine.

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

The VM clone can then be used as a drop-in replacement for the corrupted VM should a disaster condition occur.

The benefit of this backup method is that it is almost non-disruptive.

Prerequisites

You need the following items in place before you start with the procedure:

- A NFS export on your backup storage preconfigured as **rw,no_root_squash**. You will mount this export on the ESXi host. Optionally, adding the **async** option (async,rw,no_root_squash) can speed up operation but can put a high load on the NFS machine.
- The Cisco-modified ghetoVCB.sh script which can be downloaded from https://raw.github.com/gvalkov/ghettoVCB/workdir-fix/ghettoVCB.sh.
- Root access (SSH) to every ESXi host running your virtual environment.
- None of your virtual machines should have *any* snapshots created. Having snapshots will result in failure to create the backup.

Limitations and Known Issues

This approach has these limitations:

- None of your virtual machines can have snapshots created.
- You may loose the latest search indexes. If this happens, reindex all search indexes.
- The NFS storage used by Cisco WebEx Social roles will not be backed up. It's the customer's responsibility to back it up as and when appropriate. Cisco recommends that you always back up your NFS storage at nearly the same time as the Cisco WebEx Social nodes to maintain consistency.

These known issues have been identified:

- Restored VMs may reboot once the first time they are started. After that start-up should continue trouble-free.
- A file system journal recovery will need to complete following the restored VM's first boot.

Configuring the Backup

Note The procedure in this document only creates backups of Cisco WebEx Social nodes that store irrecoverable data (JSON Store, Analytics Store, RDBMS Store, Director). The rest of the Cisco WebEx Social nodes do not need to be backed up because they can be simply redeployed and rebuild their data (if any) based on data from the restored nodes. The steps in Restoring a Backup explain when to redeploy the latter nodes.

This procedure will not back up the NFS storage used by Cisco WebEx Social roles. Cisco highly recommends that you always back up your NFS storage at nearly the same time as the Cisco WebEx Social nodes to maintain consistency.

Complete the steps in these sections in succession to configure your backups:

1. Configure the ESXi Hosts, page 3
2. Configure the ghettoVCB.sh Script, page 5
3. Create Cron Jobs, page 6

Configure the ESXi Hosts

Complete these steps:

Step 1 From the vSphere Client for each ESXi host configure NTP.

a. From the vSphere Client Home screen, select **Inventory > Hosts and Clusters**.

b. Select the host you want to configure and click the **Configuration** tab.

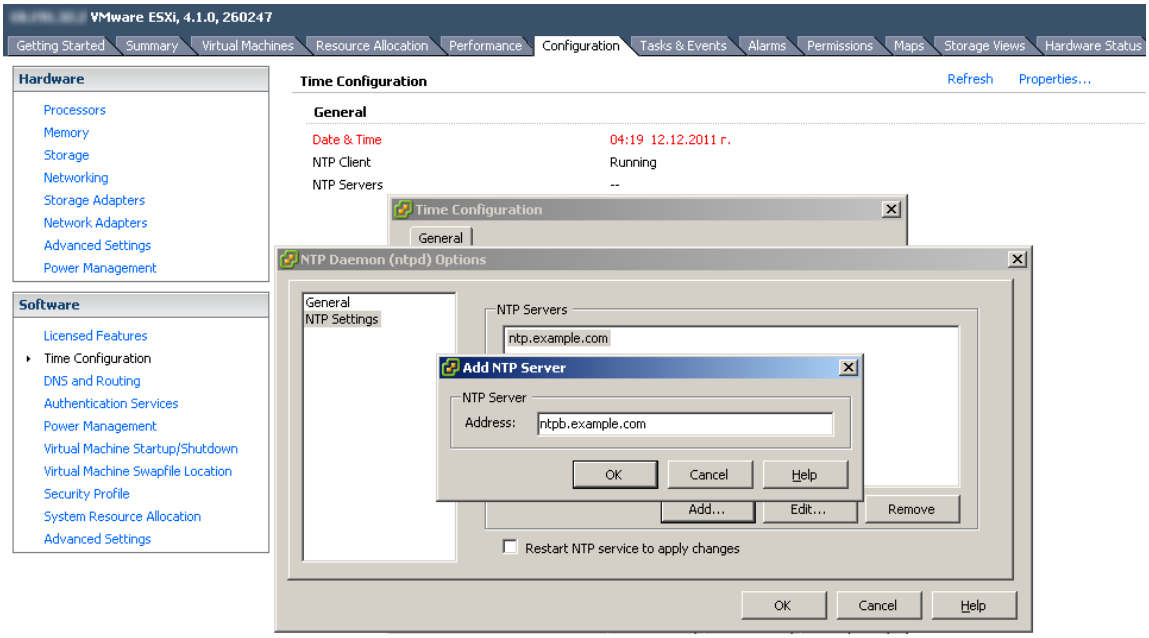
c. Click **Software > Time Configuration**.

d. Click **Properties...**.

e. Under NTP Configuration, click **Options**.

f. On the **NTP Settings** page, add your NTP server.

g. On the **General** page, select the startup policy you want and ensure the service is running.



Step 2 From the vSphere Client for each ESXi host add your NFS backup storage.

a. From the vSphere Client Home screen, select **Inventory > Hosts and Clusters**.

b. Select the host you want to configure and click the **Configuration** tab.

c. Click **Hardware > Storage**.

d. Click **Add Storage...**.

e. From the wizard's first page, select Network File System.

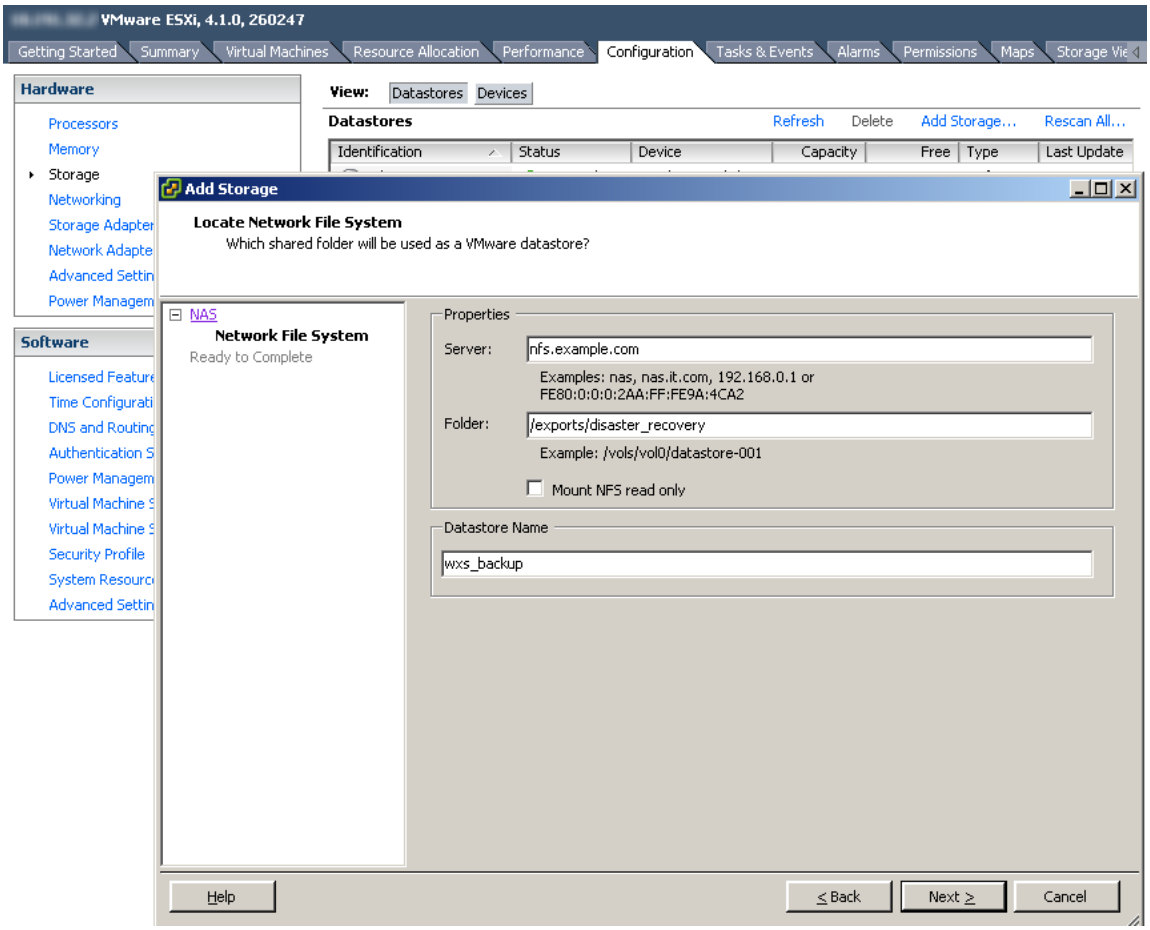
f. Specify your **Server** URL.

g. In **Folder**, specify the name of the export which you prepared to store backups.

h. Specify **Datastore Name**. A directory with this name will be created automatically on your EXS host under /vmfs/volumes/. Backups will appear in this datastore.

   The following examples are used for the purposes of this document:

   - NFS server URL: nfs.example.com
   - NFS export path (the full path to the exported directory on the NFS server): /exports/disaster_recovery
   - Datastore name: wxs_backup



Step 3 From the vSphere Client for each ESXi host enable Local Tech Support and Remote Tech Support. These daemons are required for SSH access to the ESXi hosts.

a. From the vSphere Client Home screen, select **Inventory > Hosts and Clusters**.

b. Select the host you want to configure and click the **Configuration** tab.

c. Click **Software > Security Profile**.

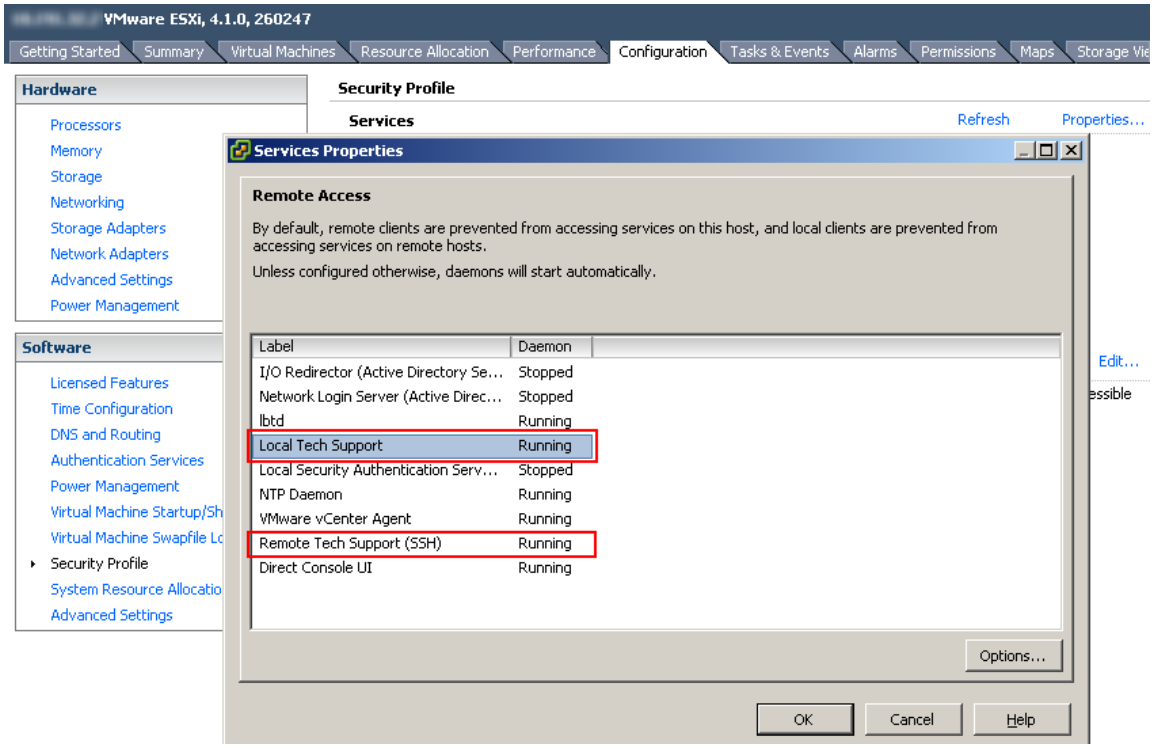
d. Click **Properties...**.

e. Select **Local Tech Support** and click Options.

f. Ensure **Start Automatically** is selected and that the service is running.

g. Select **Remote Tech Support** and click Options.

h. Ensure **Start Automatically** is selected and that the service is running.



Configure the ghettoVCB.sh Script

Next you need to copy the ghettoVCB.sh script to the backup NFS storage and configure it. Take these steps:

Step 1 Log in to the text console of any of your ESXi hosts as root.

Step 2 Go to the datastore you created in Step 2h. For example if you used the suggested datastore name (wxs_backup), the path is /vmfs/volumes/wxs_backup:

```
cd /vmfs/volumes/wxs_backup
```

Replace wxs_backup with your datastore name (case sensitive) if you used a different name.

Step 3 Create these two subdirectories: **scripts**, **vm-backup**:

```
mkdir scripts vm-backup
```

Step 4 Copy the ghettoVCB.sh script to the **scripts** directory.

Step 5 Ensure the ghettoVCB.sh script is executable:

```
chmod +x /vmfs/volumes/wxs_backup/scripts/ghettoVCB.sh
```

Replace wxs_backup with your datastore name (case sensitive) if you used a different name.

Step 6 Edit ghettoVCB.sh to match your environment and preferences:

a. Specify the local path to the vm-backup directory you created:

```
VM_BACKUP_VOLUME=/vmfs/volumes/wxs_backup/vm-backup
```

Replace wxs_backup with your datastore name (case sensitive) if you used a different name.

b. Ensure "Quiesce guest file system" is disabled:

```
VM_SNAPSHOT_QUIESCE=0
```

c. Set EMAIL_LOG to 1 to receive the script output on your email and then configure your email preferences:

```
EMAIL_LOG=1
EMAIL_SERVER=smtp.example.com
EMAIL_FROM=wxs-backup@example.com
EMAIL_TO=wxs-admin@example.com
```

Create Cron Jobs

Configure cron to run the backups:

Step 1 On each ESXi host create a cron job for each virtual machine of these types: JSON Store, Analytics Store, RDBMS Store, or Director.

a. Create a copy of your root crontab (`/var/spool/cron/crontabs/root`):

```
cp /var/spool/cron/crontabs/root /var/spool/cron/crontabs/root.b
```

b. For each VM, add a line like this *to the copy of the root crontab (root.b)*. Use the following syntax:

```
00 03 * * * /vmfs/volumes/wxs_backup/scripts/ghettoVCB.sh -m "VM Name" -w
/tmp/workdir.N > /dev/null 2>&1
```

Where:

00 03 * * * (the first five fields) are the time and date on which to start the cron job. See the `crontab(5)` man page for details on how to set these up. **These fields need to be the same for all VMs.**

VM Name is the VM name exactly as it appears in ESXi, case sensitive.

N is a unique number that helps create a dedicated working directory for each VM.

Note Use :w! or :x! in vi when saving the file to circumvent the read-only protection of root.b.

For example, if your VMs are named JSON Store, Analytics Store, RDBMS Store, and Director, you need to add these four cron lines:

```
00 03 * * * /vmfs/volumes/backup/scripts/ghettoVCB.sh -m "JSON Store" -w
/tmp/workdir.1 > /dev/null 2>&1
00 03 * * * /vmfs/volumes/backup/scripts/ghettoVCB.sh -m "Analytics Store" -w
/tmp/workdir.2 > /dev/null 2>&1
```

```
00 03 * * * /vmfs/volumes/backup/scripts/ghettoVCB.sh -m "RDBMS Store" -w
/tmp/workdir.3 > /dev/null 2>&1
00 03 * * * /vmfs/volumes/backup/scripts/ghettoVCB.sh -m "Director" -w /tmp/workdir.4
> /dev/null 2>&1
```

c. Stop cron:

```
/bin/kill $(cat /var/run/crond.pid)
```

d. Overwrite the root crontab with the root.b crontab:

```
mv /var/spool/cron/crontabs/root.b /var/spool/cron/crontabs/root
```

e. Restart cron:

```
/bin/busybox crond
```

Step 2 Make the cron configuration persistent by appending the lines you already added to root.b to your /etc/rc.local file (edit as appropriate to reflect your paths and start time):

```
#Backup the current cron configuration
cp /var/spool/cron/crontabs/root /var/spool/cron/crontabs/root.b
#Add new configuration
echo "
00 03 * * * /vmfs/volumes/backup/scripts/ghettoVCB.sh -m "JSON Store" -w /tmp/workdir.1 >
/dev/null 2>&1
00 03 * * * /vmfs/volumes/backup/scripts/ghettoVCB.sh -m "Analytics Store" -w
/tmp/workdir.2 > /dev/null 2>&1
00 03 * * * /vmfs/volumes/backup/scripts/ghettoVCB.sh -m "RDBMS Store" -w /tmp/workdir.3 >
/dev/null 2>&1
00 03 * * * /vmfs/volumes/backup/scripts/ghettoVCB.sh -m "Director" -w /tmp/workdir.4 >
/dev/null 2>&1" >> /var/spool/cron/crontabs/root.b
#Stop cron
/bin/kill $(cat /var/run/crond.pid)
#Overwrite the root crontab with the root.b crontab
mv /var/spool/cron/crontabs/root.b /var/spool/cron/crontabs/root
#Restart cron
/bin/busybox crond
```

Creating a Backup

A backup is created automatically each time the date and hour configured in cron comes. You cannot trigger a backup manually.

Once the backup has completed, you receive an informational email from the ghettoVCB scrip for each virtual machine you are backing up. Verify that the emails contain success messages for each of the backed up machines as shown below:

```
2012-11-01 10:06:37 -- info: Backup Duration: 24.88 Minutes
2012-11-01 10:06:37 -- info: Successfully completed backup for Director!
2012-11-01 10:06:38 -- info: ###### Final status: All VMs backed up OK! ######
```

Note The log is also available on the ESXi hosts that ran the script under /tmp. Look for files named ghettoVCB-*date_time*.log. Because virtual machines can be dynamically moved across ESXi hosts for better resource utilization you may need to try several hosts before you find the one storing the log file.

Example: /tmp/ghettoVCB-2012-10-30_15-43-47-41287445.log

The ghettoVCB log files are small in size but you may want to clear older logs once in a while to prevent the disk from filling up.

Restoring a Backup

Follow these steps to restore a broken Cisco WebEx Social deployment:

Step 1 Ensure the Cisco WebEx Social environment that needs restoring is devoid of any virtual machines.

a. From the vShere Client Home screen, select **Inventory > VMs and Templates**.

b. Expand your Cisco WebEx Social environment.

c. Right-click each VM and select **Remove from Inventory**.

Note You need to power-off the VM before you can remove it from inventory.

Step 2 Import the cloned VMs.

a. From the vShere Client Home screen, select **Inventory > Datastores**.

b. Right-click your backup NFS store and select **Browse Datastore**.

c. Browse to each of the backed up VMs, right-click the .vmx file and click **Add to Inventory**.

d. Follow the wizard to select where you want to import the VM.

Step 3 Start the restored RDMBS Store VM:

a. From the vSphere Client, select the VM and click **Power On**.

b. If you are presented with the “This virtual machine might have been moved or copied” alert answer “I moved it”.

Note You may need to go to the Summary tab and click “I moved it” from there.

Step 4 Start the restored Director VM.

Step 5 Delete any existing security certificates on the Director for nodes you did *not* backup.

a. Use ssh to access the Director node as admin.

b. For each node you did *not* backup, run these commands:

```
sudo salt-key -d <node FQDN>; sudo service salt-master restart
sudo puppetca --clean <node FQDN>; sudo service puppetmaster restart
```

Where *node FQDN* is the fully qualified domain name of the node.

Step 6 Start all JSON Store and Analitycs Store VMs.

a. Select each VM and click **Power On**.

b. If you are presented with the “This virtual machine might have been moved or copied” alert answer “I moved it”.

Step 7 Verify that the JSON Store replica set is running:

a. Use ssh to access any of the JSON Store nodes as admin.

b. Verify that the mongod-jsonstore service is running:

```
sudo service mongod-jsonstore status
```

c. Connect to the mongo console:

```
mongo --port 27000
```

d. Once you are in the mongo console, run this command:

```
rs.status()
```

e. In the output, look for "stateStr" lines and verify that:

- If you have a single node in the set, there is a single "stateStr" line with the value of "PRIMARY".
- If you have two nodes in the set, the respective "stateStr" lines have these values: "PRIMARY", "ARBITER", and "SECONDARY" (as shown in the example below).

```
MongoDB shell version: 2.0.3
connecting to: 127.0.0.1:27000/test
PRIMARY> rs.status()
{
        "set" : "jsonstore",
        "date" : ISODate("2012-07-03T14:22:10Z"),
        "myState" : 1,
        "members" : [
                {
                        "_id" : 0,
                        "name" : "json.site1.example.com:27000",
                        "health" : 1,
                        "state" : 1,
                        "stateStr" : "PRIMARY",
                        "optime" : {
                                "t" : 1341295121000,
                                "i" : 1
                        },
                        "optimeDate" : ISODate("2012-07-03T05:58:41Z"),
                        "self" : true
                },
                {
                        "_id" : 1,
                        "name" : "director.site1.example.com:27000",
                        "health" : 1,
                        "state" : 7,
                        "stateStr" : "ARBITER",
                        "uptime" : 6441,
                        "optime" : {
                                "t" : 0,
                                "i" : 0
                        },
                        "optimeDate" : ISODate("1970-01-01T00:00:00Z"),
                        "lastHeartbeat" : ISODate("2012-07-03T14:22:10Z"),
                        "pingMs" : 2
                },
                {
                        "_id" : 2,
                        "name" : "json2.site1.example.com:27000",
                        "health" : 1,
                        "state" : 2,
                        "stateStr" : "SECONDARY",
                        "uptime" : 6433,
                        "optime" : {
                                "t" : 1341295121000,
```

```
                                "i" : 1
                        },
                        "optimeDate" : ISODate("2012-07-03T05:58:41Z"),
                        "lastHeartbeat" : ISODate("2012-07-03T14:22:09Z"),
                        "pingMs" : 0
                }
        ],
        "ok" : 1
}
PRIMARY>
```

Step 8 Verify that the Analytics Store replica set is running:

a. Use ssh to access any of the Analytics Store nodes as admin.

b. Verify that the mongod-analyticsstore is running:

```
sudo service mongod-analyticsstore status
```

c. Run this command:

```
mongo --port 27001
```

d. Once you are in the mongo console, run this command:

```
rs.status()
```

e. In the output, look for “stateStr” lines and verify that:

- If you have a single node in the set, there is a single “stateStr” line with the value of “PRIMARY“.
- If you have two nodes in the set, the respective “stateStr” lines have these values: “PRIMARY“, “ARBITER“, and “SECONDARY” (as shown in the example below).

```
MongoDB shell version: 2.0.3
connecting to: 127.0.0.1:27001/test
PRIMARY> rs.status()
{
        "set" : "analyticsstore",
        "date" : ISODate("2012-07-05T09:53:18Z"),
        "myState" : 1,
        "members" : [
                {
                        "_id" : 0,
                        "name" : "atics.site1.example.com:27001",
                        "health" : 1,
                        "state" : 2,
                        "stateStr" : "SECONDARY",
                        "uptime" : 86237,
                        "optime" : {
                                "t" : 1341446408000,
                                "i" : 107
                        },
                        "optimeDate" : ISODate("2012-07-05T00:00:08Z"),
                        "lastHeartbeat" : ISODate("2012-07-05T09:53:16Z"),
                        "pingMs" : 0
                },
                {
                        "_id" : 1,
                        "name" : "director.site1.example.com:27001",
                        "health" : 1,
                        "state" : 7,
```

```
                        "stateStr" : "ARBITER",
                        "uptime" : 86245,
                        "optime" : {
                                "t" : 0,
                                "i" : 0
                        },
                        "optimeDate" : ISODate("1970-01-01T00:00:00Z"),
                        "lastHeartbeat" : ISODate("2012-07-05T09:53:17Z"),
                        "pingMs" : 2
                },
                {
                        "_id" : 2,
                        "name" : "atics2.site1.example.com:27001",
                        "health" : 1,
                        "state" : 1,
                        "stateStr" : "PRIMARY",
                        "optime" : {
                                "t" : 1341446408000,
                                "i" : 107
                        },
                        "optimeDate" : ISODate("2012-07-05T00:00:08Z"),
                        "self" : true
                }
        ],
        "ok" : 1
}
PRIMARY>
```

Step 9 Using templates, redeploy any Cisco WebEx Social roles whose virtual machines were not backed up. Consult the appropriate release of the *Cisco WebEx Social Installation and Upgrade Guide* for instructions while keeping the following point in mind:

- Keep each VM's IP address and hostname when restoring ir redeploying it
- If OVF templates are not available for the Cisco WebEx Social nodes you need to redeploy, take the OVF temaplates for the closest possible prevous release and use them. The Director will then find the descrepancy and upgrade the nodes automatically. Wait for all the nodes to be updated, log in to each of them as admin and run this command:

  service puppet debug

- Any App Store nodes must be started afters all other nodes are running

Step 10 After all the VMs have been restored or redeployed you may notice that Search is not working. To fix that, sign in to Cisco WebEx Social as an administrator, go to Account Settings > Server > Server Administration > Resources and execute Re-Index all search indexes.