Automating Sun™ Cluster 3.0 Data Service Setup

By Tom Bialaski - Enterprise Engineering

Sun BluePrints™ OnLine - February 2002

http://www.sun.com/blueprints

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95045 USA
650 960-1300

Part No.: 816-4242-10
Revision 2.0, 01/18/02
Edition: February 2002

Automating Sun™ Cluster 3.0 Software Data Service Setup

The purpose of this article is to provide tips on writing scripts that automate the data service installation and configuration process. The example used throughout this article is deploying the HA-NFS data service, whose agent is contained on the Sun™ Cluster 3.0 software Data Services CD-ROM. While this may be a simplistic case, since the application software (NFS server) is part of the Solaris™ Operating Environment (Solaris OE) that does not have to be installed separately, the example is helpful to illustrate some points about how to architect your scripts.

This article includes the following subjects to detail architecting your scripts:

- Overview
- Cluster configuration
- Tips for setting up a data server
- Sequences and dependencies
- Data service administration
- Backing out of data services

Overview

After you install the Sun Cluster 3.0 software and perform basic cluster configuration, your next task is to set up data services for the application or applications you plan to run on your cluster. This procedure consists of several steps. Some of the steps, such as creating a global filesystem, need to be performed from the command line. Others, such as creating a resource group, can be performed through the SunPlex™ Manager GUI. Since there are numerous steps involved, that require either executing complicated commands or traversing through several GUI screens, it is advantageous to capture these steps in a script that can be used to repeat the process.

Since it is prudent to test out mission critical applications on a test network before they are put into production, scripts are a valuable tool to capture what you did on the test network to make sure it is repeated consistently on the production network. Scripts are also useful in rebuilding systems that may be used for multiple testing purposes. Another reason for writing scripts is to allow lesser experienced system administrators to perform complex configuration tasks.

Cluster Configuration

The cluster configuration used in the examples presented in this article consists of two cluster nodes with Sun StorEdge™ T3 arrays as shared storage. The boot disk is mirrored on both cluster nodes with `qfe` network controllers used for the private cluster interconnect. FIGURE 1 shows these cluster components.

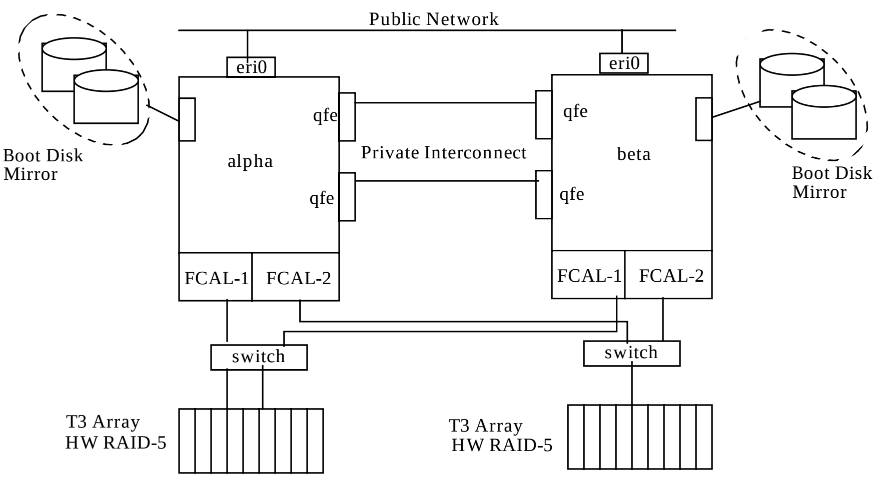


FIGURE 1 Cluster Configuration Example

The procedures used in this particular example assume the following: Sun Cluster 3.0 software and appropriate patches are already installed, the boot disk is mirrored, the quorum device is configured, and the private interconnects are set up. This is the condition that a SunTone℠ Cluster Platform is in after a few configuration questions, but these steps can be performed on any supported configuration. The two Sun StorEdge T3 arrays are configured as single RAID-5 LUNs. Not shown are the additional public network connections that should be created from unused ports on the `qfe` network controller to create Network Adapter Fail Over (NAFO) groups. For testing purposes, a NAFO group consisting of a single `eri` adapter is used.

The physical host name chosen for cluster node 1 is `alpha` and the cluster node 2 name is `beta`. These names are used throughout the examples presented. Additional explanations of Sun Cluster 3.0 software components can be obtained from previous Sun BluePrints™ Online articles, such as “Cluster Platform 220/1000 Architecture” by Enrique Vargas, published August 2001, and the *SunTone Cluster Platform User Manual*.

Tips for Data Service Setup

Before you develop your scripts, you need to determine what configuration changes are required and how those changes are performed. You need to know on which nodes specific changes need to be made and in what sequence they must be performed. Sun Cluster 3.0 software data services setup requires both *local* and *global* changes to the cluster environment. Local changes are those that are performed on each cluster node and may have to be completed on each node before some global changes are made. Global changes are made to the cluster configuration database, performed only once, and can be executed from any active cluster node.

Local Changes

Local, or per node, changes are usually performed via the Cluster Console when setting up data services in an interactive mode. However, to automate the process by creating scripts, the Cluster Console is not an option. To design a script that performs the same function you would perform during an interactive setup, you need to determine what those local changes are. The following are examples of common local changes.

- Creating NAFO groups.

  At least one NAFO group has to exist on each node before a logical hostname resource can be created. This is to assure that the data service assigned to the logical hostname will not have to fail over to another cluster node because of a single network adapter failure. The reason each node is configured separately is that NAFO groups can be created using different sets of adapters on different cluster nodes.

- Installing the data service package.

Data service agents are not automatically installed when the cluster software is installed. A set of data agents is contained on the Sun Cluster 3.0 software Data Services CD-ROM and some are provided with application software such as the iPlanet suite of products. One reason why they are not installed automatically is that it is best to load the current version when you are ready to begin using the data service and not have a potentially outdated agent installed but not being used.

- Modifying the filesystem table.

  Global filesystems are mounted using information contained in `/etc/vfstab`, similar to normal filesystem mounts. This file needs to be modified on each cluster node so either cluster node can perform the mount during system booting.

- Creating global filesystem mount points.

  Directories, where global filesystems get mounted, do not get created automatically. A directory with the same pathname needs to be created on each cluster node.

- Adding the logical host to your name service.

  Before a logical hostname resource can be created, the hostname associated with it must be resolvable through either a name service or `/etc/hosts`. If you choose not to use a name service, which eliminates an external point of failure, an entry in `/etc/hosts` needs to be created on each cluster node.

Global Changes

Global changes are made to the cluster configuration database which is automatically propagated to all cluster nodes. Therefore, these type of modifications only have to be performed from one cluster node. Examples of common global changes are:

- Creating a mirrored volume on shared storage.

  Either Solstice DiskSuite™ (SDS) or Veritas Volume Manager (VxVM) software needs to be installed on both nodes prior to performing this step.

- Creating a UFS filesystem on the shared volume.

  The `newfs` or `mkfs` command needs to be run to create the filesystem.

- Mounting the cluster filesystem.

  Once mounted on one cluster node, it automatically appears mounted on all nodes.

- Creating a resource group for the data service and activating it.

  This is a multi-step process and the last thing that needs to get done.

Sequence/Dependencies

The scripts you develop need to perform operations in the correct order to run successfully. Mapping out what the dependencies are helps you determine the sequence in which steps must take place. The following is a list of major cluster configuration tasks along with what their dependencies are.

- Shared storage configuration—If VxVM software is chosen for your volume manager, mirrored volumes can be created before the cluster software is loaded, but since VxVM software is not cluster-aware, these volumes need to be registered with the cluster *after* the cluster software is installed and configured. Solstice DiskSuite is cluster-aware and should be used to configure your shared storage after the cluster is operational.
- Data Agent package addition—Can be loaded after the Sun Cluster software core component packages are installed and while the cluster is active or inactive.
- NAFO configuration—This is not tied to any particular data agent or service. Must be configured while the cluster node is active. For testing purposes, a NAFO group consisting of only a single network adapter can be created using the system's primary network interface.
- Cluster filesystem creation—Before the filesystem can be created you must create the volumes where it will be placed. Do this with either Solstice DiskSuite or VxVM software. If VxVM software is chosen, the Veritas disk group created must be registered with the cluster before the filesystem is created.
- Cluster filesystem mounting—The mount point directory needs to be created on all nodes before it will mount. The mount operation only needs to be performed once and can be performed from any cluster node.
- Virtual hostname resource—An entry must exist in a name service or `/etc/hosts` on each node. Also, at least one NAFO group must exist on each node.
- Resource type registration—The data agent package needs to be installed on both cluster nodes. The registration command can be run on any active cluster node in the cluster.
- Resource Group creation—This can be performed on any active node once all the resources, that the group is comprised of, are created.

Script Sequence for HA-NFS Agent

A helpful technique, to determine the correct sequence to run your script commands in, is by mapping out what operations need to performed on each cluster node. FIGURE 2 shows the steps required on each cluster node to setup the HA-NFS data service. As noted in FIGURE 2, the dotted arrows show dependencies.

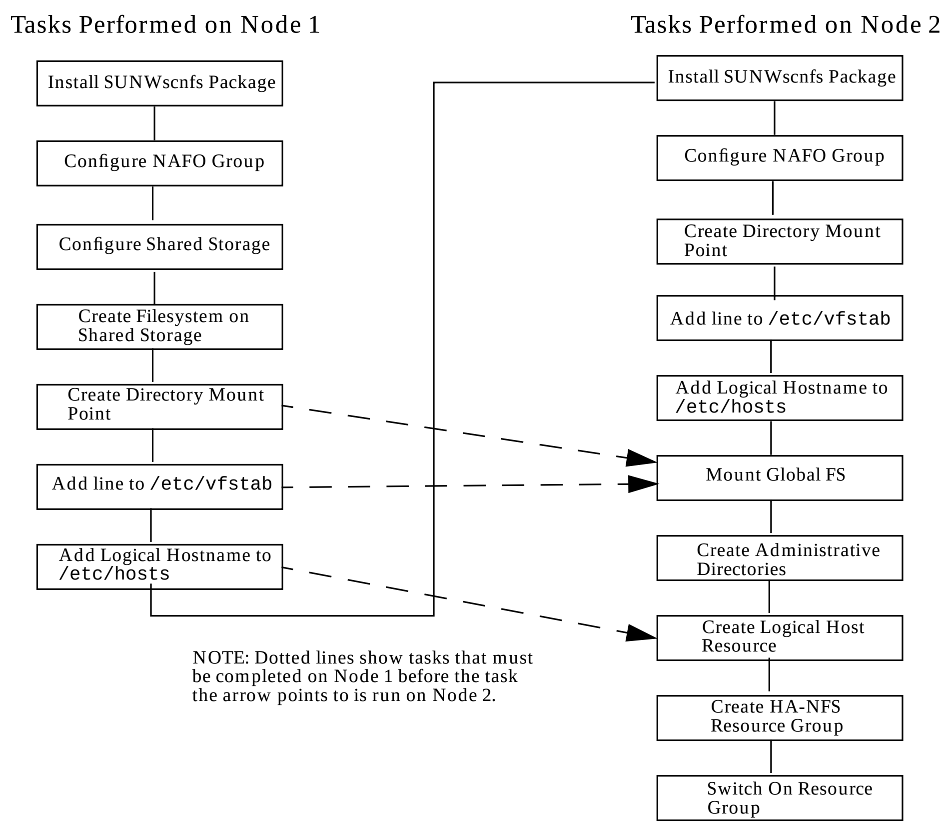


FIGURE 2 Data Service Setup Tasks and Dependencies for HA-NFS

The steps listed in FIGURE 2 above the top arrow are independent of each other and can be performed on either cluster node at any time. However, since it makes sense to execute all the commands without dependencies on the first node before a script is run on the second cluster node, FIGURE 2 presents the steps that way. This eliminates the need to go back and forth between the cluster nodes when the scripts are run.

Determining the Status of the Cluster

Included in any script should be a status check of the cluster current configuration to prevent the script from producing undesired results and to inform the user that the cluster is not in the state that the script assumes it to be in. The `scstat -n` command is a convenient way to determine whether the cluster nodes are active or online. It can also be used as a way to capture the names of the cluster nodes for use later in a script. The following is the output of the `scstat -n` command when it is run from the command line.

```
# scstat -n
-- Cluster Nodes --
                Node name       Status
                ---------       ------
  Cluster node: alpha           Online
  Cluster node: beta            Online
```

A shell script can be created to fetch the names of the cluster nodes and their current status, as shown in the following example.

```
STAT=‘scstat -n | awk ’/^  Cluster node:/ {print $3,$4}’‘
NODE1=‘echo $STAT | cut -d" " -f1‘
NODE1STAT=‘echo $STAT | cut -d" " -f2‘
NODE2=‘echo $STAT | cut -d" " -f3‘
NODE2STAT=‘echo $STAT | cut -d" " -f4‘
```

Shared Storage Configuration

When the Solaris OE initializes, all devices physically connected to the system it is running on are probed, then mapped to logical device names. Working with logical device names, in the format of *controller-target-disk-slice*, is easier than referencing the physical path to the device, which can be very cryptic. However, the order in which devices are probed is determined by which bus slot the device controller is placed in and how the device is cabled.

While differences in logical device names do not have any ill effects, they must be taken into consideration when developing automated scripts since you cannot assume logical device names are always consistent between different clusters even if the hardware components are identical. To determine what the actual physical device is by examining its logical device name the `luxadm` command is useful. The following is an example of the output from this command.

```
# luxadm inquiry /dev/rdsk/c1t0d0s0

INQUIRY:
  Physical path:
  /devices/pci@8,70000/SUNW,glc@1/fp@0,0/ssd@w50020f2,0:a,raw
Vendor:                    SUN
Product:                   T300
Revision:                  0116
Serial Number              9717E44007
Device type:               0x0 (Disk device)
...
```

From the output you can determine what type of storage device is assigned to a particular logical device name. If the shared storage is a Sun StorEdge T3 array, the associated logical device name can be extracted by examining all the device names appearing in the `/dev/rdsk` directory, searching for a product type of T300.

The following shell script routine can be used to search for the two entries in `/dev/rdsk` that match the product type of T300, the name Sun Microsystems uses to denote the Sun StorEdge T3 array.

```
find_T3() {
        ALL_DISKS=‘scdidadm -l | awk ’{print $2}’ | cut -d: -f2‘
        CANDIDATES=
        for i in $ALL_DISKS
        do
            DISKTYPE=‘luxadm inquiry ${i}s0 2>/dev/null \
| awk ’/Product:/ {print $2}’‘
            if [ "$DISKTYPE" = "T300" ]; then
                CANDIDATES="$CANDIDATES ‘basename $i‘"
            fi
        done
        NDISK=‘echo $CANDIDATES | wc -w‘
        if [ $NDISK != "2" ]; then
                gettext "ERROR: The number of T3 arrays found \
is $NDISK \n"
                gettext "Number of T3 arrays must = 2\n"
                exit 1
        fi
        T3DISK1=‘echo $CANDIDATES | awk ’{print $1}’‘
        T3DISK2=‘echo $CANDIDATES | awk ’{print $2}’‘
}
```

Once the logical device name is determined, mirrored volumes can be created using either VxVM software or Solstice DiskSuite as described in the next section.

Creating a Shared Mirrored Volume with VxVM Software

The steps for creating a shared mirrored volume with VxVM software are:

1. Initialize the disks.
2. Create a disk group.
3. Add the shared disk to the disk group.
4. Create the mirrored volume.
5. Register the disk group with the cluster.

The following commands are used to perform these steps. The variables T3DISK1 and T3DISK2 are used to represent the two Sun StorEdge T3 arrays. The values for these two variables are obtained using the code in the `find_T3()` example.

```
/etc/vx/bin/vxdisksetup -i $T3DISK1
/etc/vx/bin/vxdisksetup -i $T3DISK1
vxdg init mydskgrp $T3DISK1
vxdg -g mydskgrp adddisk $T3DISK2
vxassist -g mydskgrp make hanfsvol1 500m layout=mirror
scconf -a -D type=vxvm,name=mydskgrp,nodelist=$NODE1:$NODE2
```

After the shared storage is mirrored and registered with the cluster, you can create a filesystem on the mirrored volume as shown in the following command.

```
newfs /dev/vx/rdsk/mydiskgrp/hanfsvol1
```

Creating Shared Mirror Storage with Solstice DiskSuite

Since Solstice DiskSuite disksets, are cluster-aware the equivalent of VxVM disk groups, do not need to be registered with the cluster. However, disksets are created specifying the Disk ID (DID) numbers used to identify global devices instead of logical device names. To view all the logical device names to DID mappings, the `scdidadm` command with the `-L` option is run (see the following illustration).

```
# scdidamd -L
1 alpha:/dev/rdsk/c0t6d0 /dev/did/rdsk/d1
2 beta:/dev/rdsk/c2t0d0 /dev/did/rdsk/d2
2 alpha:/dev/rdsk/c1t0d0 /dev/did/rdsk/d2
3 beta:/dev/rdsk/c1t1d0 /dev/did/rdsk/d3
3 alpha:/dev/rdsk/c2t1d0 /dev/did/rdsk/d3
4 alpha:/dev/rdsk/c3t1d0 /dev/did/rdsk/d4
5 alpha:/dev/rdsk/c3t0d0 /dev/did/rdsk/d5
6 beta:/dev/rdsk/c0t6d0 /dev/did/rdsk/d6
7 beta:/dev/rdsk/c3t0d0 /dev/did/rdsk/d7
8 beta:/dev/rdsk/c3t1d0 /dev/did/rdsk/d8
```

Before an Solstice DiskSuite diskset can be created, the logical device names of the shared storage devices need to be translated into the equivalent DID device number. Using the function `find_T3()` (shown in a previous example), you can find the logical device names, then convert them to the equivalent DID device name by inserting the following lines into your script.

```
DID1=‘scdidadm -l | grep $T3DISK1 | awk ’{print $3}’‘
DID2=‘scdidadm -l | grep $T3DISK2 | awk ’{print $3}’‘
```

The steps for creating a mirrored volume using Solstice DiskSuite software are:

1. Create a cluster disk group specifying the potential cluster node owners.
2. Add the two Solstice DiskSuite T3 Arrays to the cluster disk group.
3. Initialize the two Solstice DiskSuite T3 Arrays.
4. Create a disk mirror.
5. Add the two Solstice DiskSuite T3 Arrays to the mirror.

The following example shows the commands that perform these operations.

```
metaset -s mydskset -a -h $NODE1 $NODE2
metaset -s mydskset -a $DID1 $DID2
metainit -s mydskset mydskset/d0 1 1 ${DID1}s0
metainit -s mydskset mydskset/d1 1 1 ${DID2}s0
metainit -s mydskset mydskset/d100 -m mydskset/d0
metattach -s mydskset mydskset/d100 mydskset/d1
metaset -s mydskset -a -m $NODE1
metaset -s mydskset -a -m $NODE2
```

You can now create the filesystem as shown in the following example.

```
newfs /dev/md/mydskset/rdsk/d100
```

Mounting the Global Filesystem

After a global filesystem is created on the mirrored shared storage devices, the `mount` command can be run from any cluster node. However, before the mount can take place, a mount directory must be created on each cluster node and the entry that performs the mount must exist in each cluster nodes's `/etc/vfstab` file. The following example shows the commands run on each cluster node to perform these operations.

On `beta`:

```
mkdir /global/nfs
echo "/dev/vx/dsk/mydskgrp/hanfsvol1 \
/dev/vx/rdsk/mydskgrp/hanfsvol1 /global/nfs ufs 2 yes \
global,logging" >> /etc/vfstab
```

On `alpha`:

```
mkdir /global/nfs
echo "/dev/vx/dsk/mydskgrp/hanfsvol1 \
/dev/vx/rdsk/mydskgrp/hanfsvol1 /global/nfs ufs 2 yes \
global,logging" >> /etc/vfstab
mount /global/nfs
```

Once the global filesystem is mounted, application software can be placed in it and administrative directory and files can be created there. For the HA-NFS example, no additional application software is required, but you need to create some additional directories and configuration files as shown in the following example.

On `beta`:

```
mount /global/nfs
mkdir /global/nfs/admin
mkdir /global/nfs/admin/SUNW.nfs
echo 'share -F nfs -o rw -d"HA-NFS" /global/nfs/data' > \
/global/nfs/admin/SUNW.nfs/dfstab.nfs-res
mkdir /global/nfs/data
chmod 777 /global/nfs/data
```

Creating the Data Service Resource Group

When a data service failure is detected on an active cluster node, the data service is transferred, or failed-over, to a working cluster node. The failover unit that Sun Cluster 3.0 software uses is the resource group, which is a collection of *resources* that are required for the data service to run. Resources are categorized by their type. For example, the resource that represents the NFS data service is of type `SUNW.nfs` and the resource for failing-over the shared storage device is of type `SUNW.HAStorage`. Before a resource can be created, you need to register its type.

A logical hostname resource is required for all data services since this name represents the IP address clients use to access the service and must be transferable from one cluster node to another. Although it is considered a resource, no resource type is associated with it. Therefore, there is no resource type that needs to be registered with the cluster.

The basic steps to create a resource group for a data service are:

1. Register the resource types of the resources used in the group, if not already registered.

2. Create a resource group specifying the cluster nodes that can run it along with the pathname of any administrative directory.

3. Add required resources to the resource group.

4. Place the resource group in the online mode.

The following example illustrates those steps.

On beta:

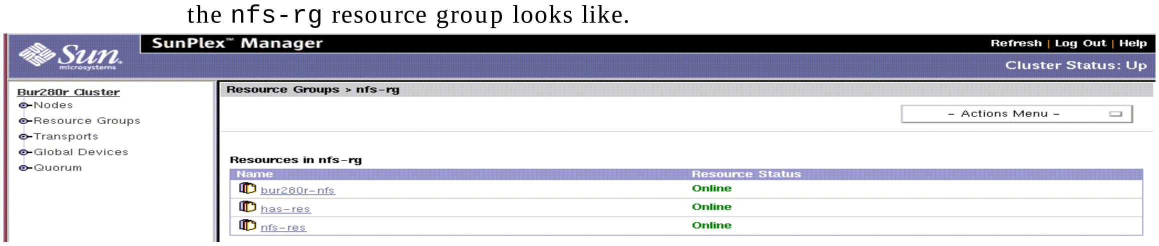
```
scrgadm -a -t SUNW.nfs
scrgadm -a -t SUNW.HAStorage
scrgadm -a -g myresgrp -h alpha,beta -y \
Pathprefix=/global/nfs/admin
scrgadm -a -L -g myresgrp -l hostname
scrgadm -a -j hares -g myresgrp -t SUNW.HAStorage -x \
ServicePaths=/global/nfs -x AffinityOn=True
scrgadm -a -j nfsres -g myresgrp -t SUNW.nfs -y \
Resource_dependencies=hares
scswitch -Z -g myresgrp
```

Data Service Administration

After the setup scripts are run, you can either manage the data service from the command line or through the SunPlex Manager GUI. By default, each cluster node is configured with an apache web server which listens to port 3000. Since a password must be provided to run the SunPlex Manager GUI, a secure port is used to encrypt the transfer of it. To invoke the SunPlex Manager GUI, enter the address of one of the cluster nodes in a web browser. For example:

```
https://alpha.sunblueprints.com:3000
```

After entering a login (`root`) and password, the cluster configuration data is displayed. To examine the HA-NFS resource group, that was setup up by the scripts, select **Resource Groups** in the left window pane. The following screenshot is what the `nfs-rg` resource group looks like.



The **Action Menu** allows you to perform common cluster administration functions like switching the service to the other cluster node. A nice feature of the SunPlex Manager GUI is that the command that is executed for a chosen action is displayed. You can use this output as a guide to developing scripts for common administration functions.

Backing Out Data Services

After a data service has been configured and initialized, you may decide to disable and de-install it. This activity is common in test or pre-production environments where cluster data services are staged before they are rolled out into production. In this section, the steps required to back out data services to restore the original configuration are detailed. The examples provided perform the operations through the command line, although the same results can be obtained through the SunPlex Manager GUI.

Removing a Resource Group

Before you can remove any resource group that you created, the resources contained in that group have to be removed. You cannot remove resources or resource groups while they are online, so they must be switched offline first. To remove the HA-NFS resource group from the command line, execute these commands from either cluster node:

```
alpha# scswitch -z -g myresgrp -h ""
alpha# scswitch -n -j nfsres
alpha# scswitch -n -j hasres
alpha# scswitch -n -j logicalhost
alpha# scrgadm -r -j nfsres
alpha# scrgadm -r -j hasres
alpha# scrgadm -r -j logicalhost
alpha# scswitch -u -g myresgrp
alpha# scrgadm -r -g myresgrp
```

Removing a Global Filesystem and Global Device Group

Global filesystems that were created for a particular data service can be removed or remain so they can be used to support other data services. If you wish to remove them along with associated cluster device groups, perform the following steps:

On either cluster node:

```
alpha# rm -r /global/nfs/admin
alpha# rm -r /global/nfs/data
(if running VxVM)
alpha# scswitch -m -D mydskgrp
alpha# scconf -r -D name=mydskgrp
(if running Solstice DiskSuite)
alpha# metaclear -s mydskset -a
alpha# metatset -s mydskset -d -h beta
alpha# metaset -s mydskset -f -d $DID1 $DID2
alpha# metaset -s mydskset -d -h alpha
```

On both cluster nodes:

```
# vi /etc/vfstab
(remove line added by install script)
# rm -r /global/nfs
```

Summary

Although the HA-NFS data service example described in this article is a simple case, the methodology presented is pertinent when creating more complicated data service setup scripts. Determining what changes need to be made to local files on each cluster node and what changes are made to the cluster environment, is key to understanding on which cluster nodes setup scripts need to run. By examining the dependencies between various cluster setup operations, you can develop the sequence of the steps required.

Any well planned setup script should check the current configuration of the cluster before it runs operations that change the configuration. Since you cannot assume that logical device names are always identical, even between similar hardware configuration, you should probe the cluster's physical devices to find out what logical device names they are assigned, rather than specifying hard-coded logical names. Tips for writing scripts that check the cluster status and probe devices were provided along with scripting examples. These can be enhanced to match your specific cluster environment.

Author's Bio: Tom Bialaski

Tom Bialaski is a Senior Staff Engineer with Sun Microsystems, Inc. He is a major contributor to the Sun BluePrints books program, authoring Solaris Guide for Windows NT Administrators, co-authoring Solaris and LDAP Naming Services and Resource Management, and writing numerous articles for Sun BluePrints Online. Tom began his career at Sun Microsystems, Inc. in 1984 as a Systems Engineer and since that time has held several technical positions.