IBM

HACMP/ES Customization Examples

Configure Lotus Domino servers for high availability

Program cluster-aware applications

Detailed, step-by-step examples

Ole Conradsen
Andrea Lucenti
Mohd Idris Abdul Rahman
Viktor Sebesteny

ibm.com/redbooks

Redbooks

SG24-4498-01

International Technical Support Organization

HACMP/ES Customization Examples

May 2000

Second Edition (May 2000)

This edition applies to Version 4, Release 3.1 of HACMP/ES and HACMP, program number 5765-D28, feature number 0885 and feature number 0882 for use with the AIX Version 4.3.1 operating system.

This document created or updated on May 31, 2000.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. JN9B Building 003 Internal Zip 2834
11400 Burnet Road
Austin, Texas 78758-3493

Contents

Figures

Tables

Preface

This redbook provides examples on how to configure RISC System/6000 and HACMP in various environments and for various applications. The examples in the book are based on AIX Release 4.3.3 and HACMP/ES Release 4.3.1. We chose to use HACMP/ES for the examples because we assume this will be the preferred platform for the future, and therefore, it makes more sense for the reader. There is, however, only minor differences in the process of customizing the HACMP Classic and the HACMP/ES versions.

The goal of this book is to give the reader some practical examples on how to customize HACMP and tailor the configuration for specific customer needs. We include two chapters with sample application programs. One example is how to configure Lotus Domino for high availability, the other example is how to write a cluster aware application in which a client application checks the cluster status before it requests a transaction. We also include a chapter about HACMP in a SAP R/3 environment.

These examples will make it faster and easier to install and configure HACMP in customer situations. The SAP R/3 chapter was created with extensive support from SAP in Waldorf and IBM Germany. The SAP configurations are recognized by SAP. This means that if SAP is running in a HACMP environment, and the configuration is as described in Chapter 8, “HACMP and SAP R/3” on page 109, it will be easier to get support from SAP. In general, we describe “best practice” in HACMP configuration, and if the examples are followed, it will provide a smoother and faster implementation in most customer installations.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Austin Center.

Ole Conradsen is a Project Leader at the International Technical Support Organization, Austin Center. He writes extensively about HACMP, HAGEO, Lotus Domino, and e-business solutions. Before joining the ITSO, he worked with AIX projects in IBM Denmark for 11 years. Ole holds a Master of Science degree in Electrical Engineering from The Technical University of Denmark.

Andrea Lucenti is an IT Specialist in Italy. He has two years of experience in the AIX high availability field. He holds a Ph.D. in Physics from the University of Milan. His areas of expertise include support and solution designing in AIX, HACMP, and TSM for Strategic Outsourcing.

Mohd Idris Abdul Rahman is an IT Specialist in Malaysia. He has six years of experience in the RS/6000 and AIX high availability fields. He has worked at IBM for three years. His areas of expertise include AIX, HACMP, and RS/6000 SP. Idris holds a BSEE degree from Worcester Polytechnic Institute, Massachusetts.

Viktor Sebesteny is an IT Specialist in Hungary. He has been working for IBM for four years with AIX and RS/6000. He has a Bachelor of Science degree in Electrical Engineering. His areas of expertise and responsibility include the support of AIX, HACMP, and high-end SP systems.

Thanks to the following people for their invaluable contributions to this project:

Michael Coffey
IBM Poughkeepsie

John Easton
IBM High Availability Cluster Competency Centre (UK)

Special thanks to the editors for their help in finalizing the text and publishing the redbook:

Elizabeth Barnes
John Owczarzak
Milos Radosavljevic
International Technical Support Organization, Austin Center

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Please send us your comments about this or other Redbooks in one of the following ways:

- Fax the evaluation form found in “IBM Redbooks review” on page 321 to the fax number shown on the form.
- Use the online evaluation form found at `http://www.redbooks.ibm.com/`
- Send your comments in an Internet note to `redbook@us.ibm.com`

Chapter 1. New functions in HACMP/ES 4.3.1

This chapter addresses the new functionality in HACMP/ES 4.3.1. In this chapter, we list all the new features and detail those that affect the cluster configuration and management. In the following chapters, we further explore these new features.

1.1 New features in HACMP/ES 4.3.1

New features have been introduced in HACMP/ES 4.3.1 to help you in planning, configuring, and maintaining the cluster. These features are:

- The Cluster Resource Manager. A new facility that centralizes the storage of, and publishes updated information about, the content, location, and status of HACMP/ES-defined resource groups.
- Enhanced HAView capability. You can use HAView to monitor resource group content, status, and location. HAView receives updated resource group information from the Cluster Resource Manager.
- Ability to perform DARE resource group migrations using the SMIT interface.
- Ability to change the location of any HACMP/ES log file to a different directory through SMIT.
- Ability to use ATM hardware address swapping functionality on adapters that are on the same ATM switch. This functionality applies to both regular and ATM LAN emulation environments.
- Integration of the AIX Fast Connect application, making it easy to configure as a highly available resource in the event of node failure.
- Integration of Communications Server for AIX. Allows you to configure CS/AIX DLC profiles as highly available resources.
- Enhanced C-SPOC functionality for creating shared volume groups and file systems and for adding disk definitions or removing them to/from a cluster.
- AIX Automatic Error Notification. A feature that provides automatic configuration of AIX Error Notification, greatly simplifying the defining of error notify methods in HACMP.
- Error Log Emulation functionality. Allows you to emulate an error in order to test whether your pre-specified response action (notify method) takes place as planned.

- Ability to migrate from HACMP for AIX High Availability Subsystem to HACMP/ES on a node-by-node basis without bringing the cluster down.
- Ability to swap a service/boot IP address to a specified available standby adapter on the same node and network, allowing you to change hot pluggable adapters without powering off the node. This dynamic adapter swap can be performed through SMIT. However, if you are running HACMP on a system that allows hot-plug removal and replacement of adapters, special care must be taken when replacing network adapters.
- Web-based online planning worksheets, included in the samples directory, that automatically configure your cluster after guiding you through the planning process.

In the following sections, we describe and test some of these new features.

1.2 Resource group migrations using SMIT

The Dynamic Reconfiguration (DARE) Resource Migration utility allows you to change the status or location of a resource group without stopping cluster services. Before performing a resource migration, you must choose a sticky or non-sticky migration.

A sticky migration permanently attaches a resource group to a specified node. The resource group does not come back to the default node after a node fallover or reintegration. The sticky designation supersedes all other resource group location policies; the new sticky location becomes the highest priority node.

Resource groups on nodes not designated sticky are, by default, non-sticky. Non-sticky resource groups are temporarily placed on the specified node until the next fallover or reintegration occurs, and all resource group location policies are reevaluated.

The Dynamic Reconfiguration Resource Migration utility provides improved cluster management by allowing you to:

- Bring a resource group up/online (this option also restores a resource group to its default setting, removing any sticky markers).
- Bring a resource group down/offline.
- Move a resource group to a new location.
- Perform maintenance on a node without losing access to the node’s resources.
- Relocate resource groups for enhanced performance.

Using `cldare` from the command line, you can change the status or location of multiple resource groups on multiple nodes at the same time; with the SMIT interface, you can perform only one migration at a time. You can also disable resource groups dynamically, preventing them from being acquired during a reintegration. This disabling option allows a “swap” of resources in certain situations.

To access the DARE migration functions using the SMIT interface, choose **Cluster System Management -> Cluster Resource Group Management** from the main HACMP menu.

Here, SMIT presents the following options for dynamic resource migration.

For each migration option, SMIT offers the choice to emulate the migration rather than actually performing it.

Bring a Resource Group Online: This is equivalent to using the `cldare default` keyword to start a resource group. Choose this to activate all resources in a specified resource group on the node designated as highest priority for that resource group. You can also use this option to return a resource group to its default setting, removing any previously set sticky designation.

Bring a Resource Group Offline: This is equivalent to the `cldare stop` keyword. Use this option to deactivate all resources in a specified resource group. You can choose sticky or normal migration.

Move a Resource Group: This option allows you to deactivate a resource group on one node and then activate it on another node. You can choose sticky or normal migration.

1.3 Change the location of any HACMP/ES log file

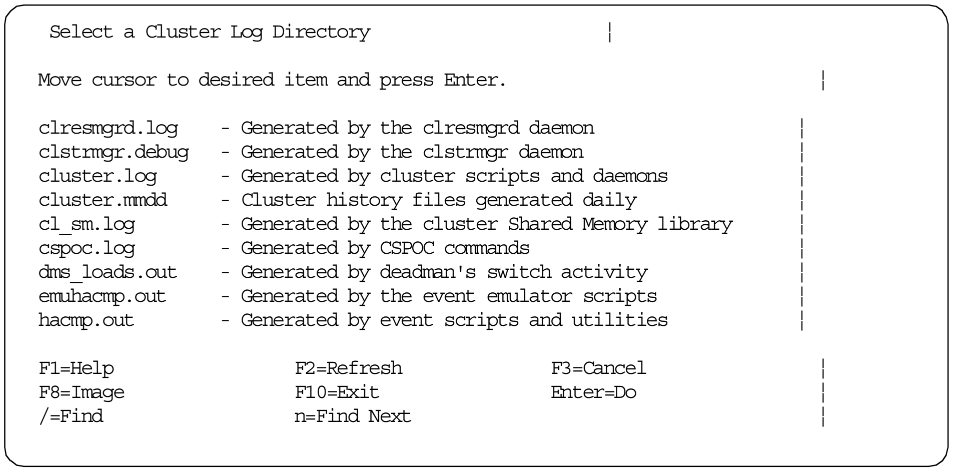
You can redirect a cluster log from its default directory to a directory of your choice. Should you redirect a log file to a directory of your choice, keep in mind that most logs can grow up to 2 MB. However, hacmp.out and clresmgrd.log can become bigger, and 14 MB allocated disk space is recommended for them.

Note

Logs should not be redirected to shared file systems or NFS file systems. Though this may be desirable in rare cases, such action may cause problems if the file system needs to unmount during a fallover event.

To redirect a cluster log from its default directory to another destination, complete the following steps:

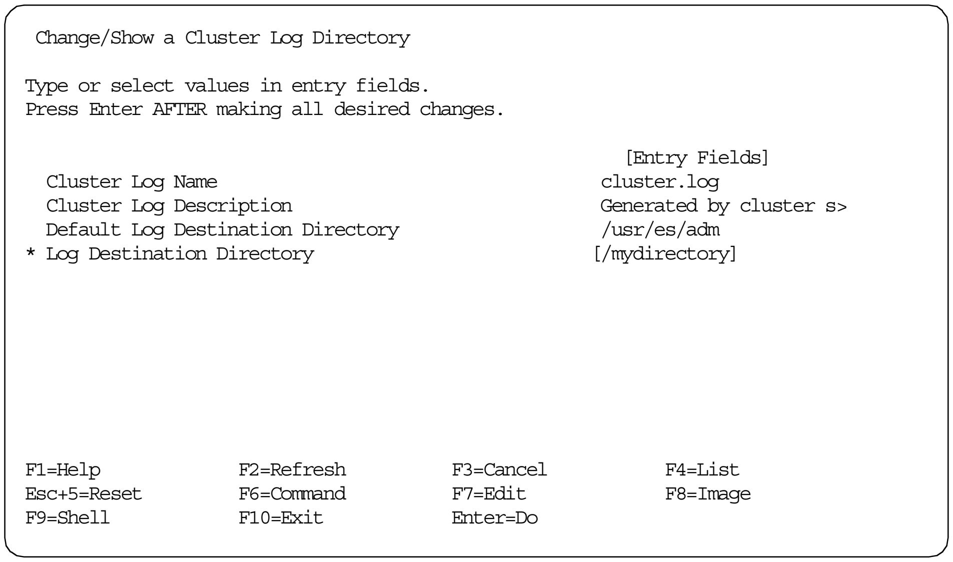
1. Enter: `smitty hacmp`
2. Select **Cluster System Management -> Cluster Log Management -> Change/Show Cluster Log Directory.**



```
  Select a Cluster Log Directory                     |

 Move cursor to desired item and press Enter.                               |

 clresmgrd.log    - Generated by the clresmgrd daemon                     |
 clstrmgr.debug   - Generated by the clstrmgr daemon                      |
 cluster.log      - Generated by cluster scripts and daemons              |
 cluster.mmdd     - Cluster history files generated daily                 |
 cl_sm.log        - Generated by the cluster Shared Memory library        |
 cspoc.log        - Generated by CSPOC commands                           |
 dms_loads.out    - Generated by deadman's switch activity                |
 emuhacmp.out     - Generated by the event emulator scripts               |
 hacmp.out        - Generated by event scripts and utilities              |

 F1=Help                F2=Refresh             F3=Cancel                    |
 F8=Image               F10=Exit               Enter=Do                     |
 /=Find                 n=Find Next                                         |
```

Figure 1. SMIT panel to select HACMP/ES log files and change their location

3. Select a log that you want to redirect, for example, cluster.log, as shown in Figure 2 on page 5.



```
 Change/Show a Cluster Log Directory

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
  Cluster Log Name                                     cluster.log
  Cluster Log Description                              Generated by cluster s>
  Default Log Destination Directory                    /usr/es/adm
* Log Destination Directory                           [/mydirectory]






F1=Help             F2=Refresh          F3=Cancel           F4=List
Esc+5=Reset         F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 2. SMIT panel to select log file location

4. Edit the last field and choose the destination directory (/mydirectory in this example). After you change a log directory, a prompt appears reminding you to synchronize cluster resources from this node (cluster log ODMs must be identical across the cluster). The cluster log destination directories as stored on this node will be synchronized to all nodes in the cluster. Log destination directory changes will take effect when you synchronize cluster resources, or if the cluster is not up, the next time cluster services are restarted.

1.4 Enhanced C-SPOC functionality

HACMP/ES provides the Cluster-Single Point of Control (C-SPOC) utility that simplifies maintenance of shared LVM components in clusters of up to 32 SP nodes in a single partition, or eight nodes in clusters that include stand-alone RS/6000s or that span partitions on an SP. C-SPOC commands provide comparable functions in a cluster environment to the standard AIX commands that work on a single node. For example, the C-SPOC utility includes a command called `cl_chlv` that provides similar functions to the AIX `chlv` command. (The C-SPOC command calls the AIX command.) By automating repetitive tasks, C-SPOC eliminates a potential source of errors and speeds up the process. LVM maintenance tasks that you can perform using C-SPOC are as follows:

- Shared volume groups
  - Create a shared volume group
  - Import a volume group
  - Extend a volume group
  - Reduce a volume group
  - Mirror a volume group
  - Unmirror a volume group
  - Synchronize volume group mirrors
  - List all shared volume groups
  - List all active shared volume groups
  - Display characteristics of a shared volume group
- Shared logical volumes
  - Create a logical volume
  - Make a copy of a logical volume
  - Remove a copy of a logical volume
  - List all shared logical volumes by volume group
  - Change or view the characteristics of a shared logical volume (name, size)
  - Remove a shared logical volume
- Shared file systems
  - Create a shared file system
  - List all shared file systems
  - Change or view the characteristics of a shared file system
  - Remove a shared file system
- Physical volumes
  - Add a definition to cluster nodes
  - Remove a definition from cluster nodes

Note

The C-SPOC commands only operate on both shared and concurrent LVM components that are defined as part of an HACMP/ES resource group.

In Chapter 2, “Configuration management in your cluster” on page 25, we show how C-SPOC features work in actual scenarios and we supply some examples.

When you execute a C-SPOC command, the utility determines on which node to perform the operation and then executes the required commands on that node. Typically, C-SPOC executes the command on the node that owns the LVM component (has it varied on). However, you can use C-SPOC commands (on the command line, not from SMIT) on an LVM component that is not currently activated on any cluster node. In this case, C-SPOC determines which node will own the LVM component when it is activated, as specified for the HACMP/ES resource group, and performs the operation on that node.

The C-SPOC commands that modify LVM components, such as `cl_chlv`, require that you specify a resource group name as an argument. The LVM component that is the target of the command must be configured in the resource group specified. C-SPOC uses the resource group information to determine on which nodes it must execute the operation specified.

When removing a file system or logical volume using the C-SPOC `cl_rmfs` and `cl_rmlv` commands, the target file system or logical volume must not be configured as a resource in the resource group specified. You must unconfigure it from the resource group before removing it.

When you change the definition of a shared LVM component in a cluster, the operation updates the LVM data that describes the component on the local node and in the Volume Group Descriptor Area (VGDA) on the disks in the volume group. AIX 4.3 LVM enhancements allow all nodes in the cluster to be aware of changes to a volume group, logical volume, and file system at the time the changes are made, rather than waiting for the information to be retrieved during a lazy update.

If for some reason the node is not updated via the C-SPOC enhanced utilities due to an error condition (a node is down, for example), the volume group will be updated and the change will be taken care of during the “lazy update” mechanism or during execution of the `clvaryonvg` command. If node failure does occur during a C-SPOC operation, an error is displayed on the screen and the error messages are recorded in the C-SPOC error log (/tmp/cspoc.log is the default location of this log). Error reporting provides detailed information about inconsistencies in volume group state across the cluster. If this happens, you must take manual corrective action.

1.5 Automatic error notification and error emulation

Using a SMIT screen option, you can configure error notification automatically for the cluster resources listed below, list currently defined automatic error notify entries for the same cluster resources, or remove previously configured automatic error notify methods. Before you configure automatic error notification, you must have a valid HACMP/ES configuration.

Note

Automatic error notification should be configured only when the cluster is not running.

Choosing to add error notify methods automatically runs the `cl_errnotify` utility, which turns on error notification on all nodes in the cluster for the following devices:

- All disks in the rootvg volume group
- All disks in HACMP/ES volume groups, concurrent volume groups, and file systems.
- The SP Switch adapter

Automatic error notification applies to selected hard, non-recoverable error types: disk, disk adapter, and SP Switch adapter errors. No media errors, recovered errors, or temporary errors are supported by this utility.

Executing automatic error notification assigns one of two error notification methods for all the error types noted:

- cl_failover is assigned if a disk or an adapter (including the SP Switch adapter) is determined to be a single point of failure and its failure should cause the cluster resources to fall over. In case of a failure of any of these devices, this method logs the error to hacmp.out and shuts down the cluster software on the node. It first tries to do a graceful shutdown with takeover; if this fails, it calls cl_exit to shut down the node.
- cl_logerror is assigned for all other error types. In case of a failure of any of these devices, this method logs the error to hacmp.out. You can also use the utility to list currently defined auto error notification entries in your HACMP/ES cluster configuration and to delete all automatic error notify methods.

In Chapter 13, “AIX Error Notification and device monitoring” on page 205, we demonstrate how to implement error notification in an actual scenario.

1.6 Swap service or boot address to an available standby adapter

You can use the dynamic adapter swap feature to swap the IP address of an active service or boot adapter with the IP address of an active, available standby adapter on the same node and network. Cluster services do not have to be stopped to perform the swap, and the network down time is minimized. After the adapter swap, the service/boot adapter becomes an available standby adapter. This feature is effective when you have a hot-pluggable adapter. In this case, HACMP/ES makes the adapter unavailable as a standby when you pull it from the node. When the new adapter card is placed in the node, the adapter is incorporated into the cluster as an available standby again. Then, you can use the dynamic adapter swap feature to swap the IP address from the standby back to the original adapter. Otherwise, HACMP/ES will configure the service/boot and standby address on their original adapters when cluster services are restarted after a cluster stop. Indeed, HACMP/ES does not record the swapped adapter information in the AIX ODM.

However, if you are running HACMP on a system that allows hot plug removal and replacement of adapters, special care must be taken when replacing network adapters that are part of an active HACMP cluster configuration.

First, determine the interface name of the adapter you want to remove and later replace. Then issue the following command replacing the string `INAME` with the interface name you want to remove and record the IP address displayed:

```
lsattr -El ‘INAME’ -a netaddr -F value
```

Then issue the `netstat -i` command and verify that the recorded IP address is not configured on any other interface. If it is, you will have to stop HACMP on this node before the adapter is *hot-replaced.* This can be done by issuing the following command:

```
/usr/sbin/cluster/utilities/clstop -y -N -g
```

Once the node node_down event is complete, return the interface to the startup state and execute the following command:

```
/etc/rc.net-boot
```

If the recorded IP address is not currently configured on the node, you can remove and replace the adapter without taking HACMP down on the node.

Note

The dynamic adapter swap feature is not supported on the SP Switch network.

The dynamic adapter swap feature can only be performed within a single node. You cannot swap the IP address of a service or boot address with the IP address of a standby adapter on a different node. To move an IP address to another node, move its resource group using the DARE Resource Migration utility.

1.7 Web-based online planning sheets

This section covers how to use the Web-based online worksheets provided with the HACMP/ES software in the /usr/es/lpp/cluster/samples/worksheets directory. The online worksheet program is a tool to aid you in planning and entering your cluster configuration and then applying the configuration directly to your cluster. To use the online worksheets, you must transfer the worksheet program files to a PC-based system equipped with the appropriate Web browser.

Web-based planning is a online counterpart to the paper worksheet. The information may be consolidated or ordered in a slightly different way than the paper worksheets, but the information is basically the same. The major enhancement is the fact that the online worksheet can produce an AIX HACMP configuration file. This configuration file can be loaded into the system and the configuration of your cluster is made for you.

1.7.1 Installing the worksheet program

To use the worksheets, you need a PC-based system running an up-to-date version of Microsoft Internet Explorer (Version 4.0 or higher) Web browser. The installation steps are as follows:

1. FTP (binary mode) to your PC the following two files: worksheets.html and worksheets.jar placed in the /usr/es/lpp/cluster/samples/worksheets directory after installation of the cluster.adt.es.client.samples.demos fileset.
2. Set the CLASSPATH variable to the worksheets.jar pathname:
   a. Windows 95: You must edit the autoexec.bat file to include the CLASSPATH variable:

      ```
      set CLASSPATH=c:\< folder name>\worksheets.jar
      ```

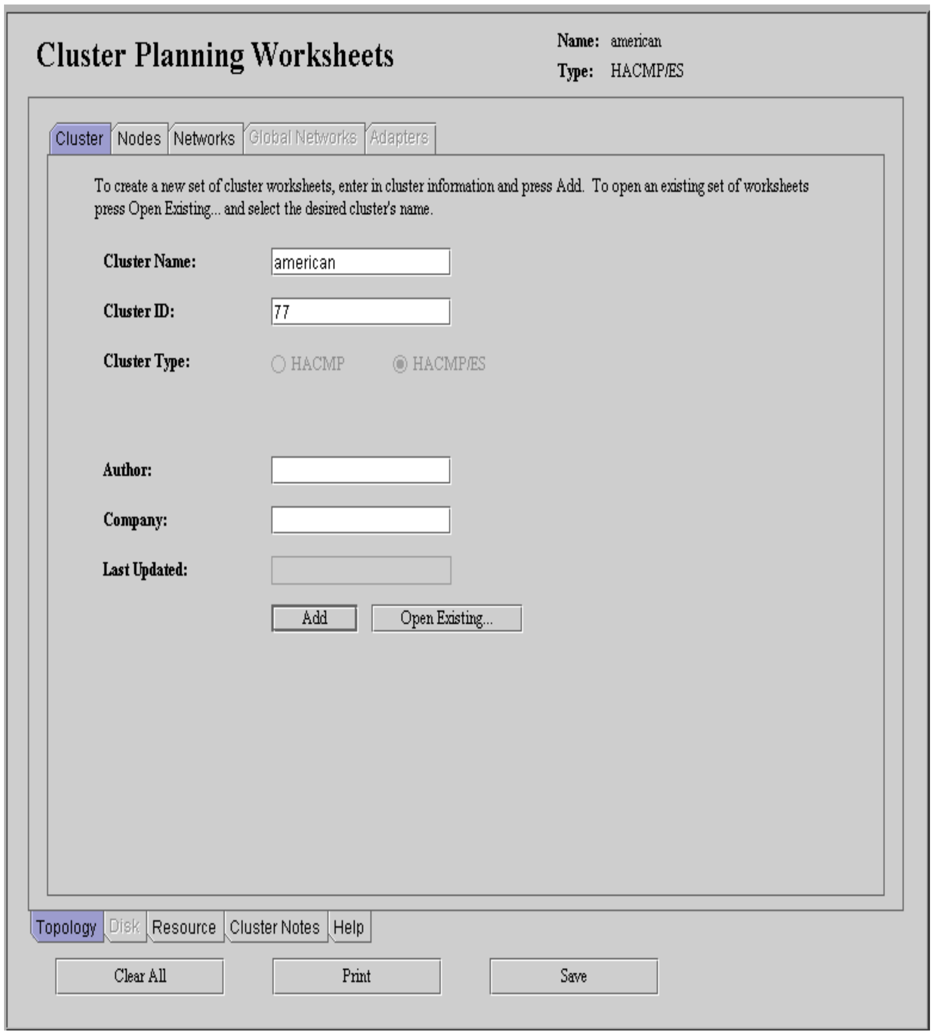
b. Windows NT**:** Go to **Control Panel -> System Properties -> Environment**. Add CLASSPATH as the variable and the full pathname for worksheets.jar as the value.

3. In your browser, open worksheets.html.

1.7.2 Starting the planning

The first panel, shown in Figure 3, asks you the cluster name and the cluster ID or to open an existing set of worksheets.

.



Cluster Planning Worksheets

Name: american
Type: HACMP/ES

To create a new set of cluster worksheets, enter in cluster information and press Add. To open an existing set of worksheets press Open Existing... and select the desired cluster's name.

Cluster Name: american

Cluster ID: 77

Cluster Type: HACMP [✓ HACMP/ES]

Author:

Company:

Last Updated:

Figure 3. Online worksheet: Starting panel

The bottom tabs indicate the three major planning topics. As you select a bottom tab, a series of top subtopic tabs are enabled. At any time, the base panel buttons allow you to do the following actions:

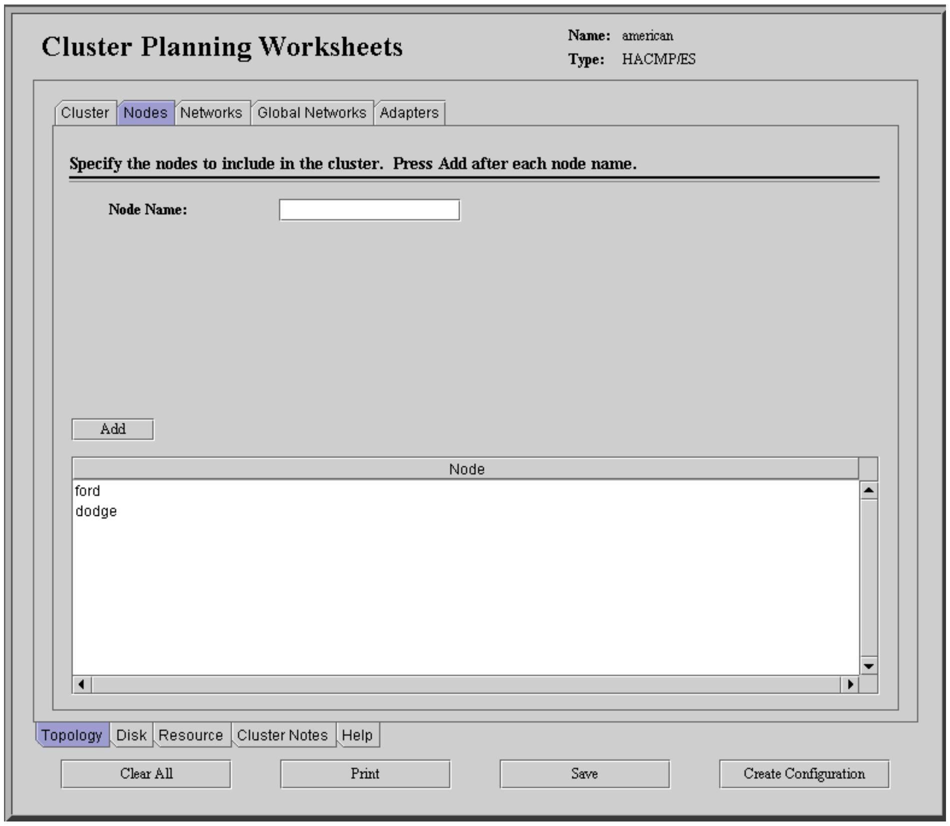
Clear All	Deletes the entire set of worksheets you have open.
Print	Prints all of the configuration data you have entered thus far for the set of the work sheets you have open.
Save	Writes the cluster configuration entered thus far to the PC disk to a specified worksheet (.ws) file.
Create Configuration	Creates the AIX file that you will transfer to an AIX cluster node to apply the configuration data to the cluster.

Important

The browser refresh or reload buttons clear the panel of the information you have entered.

1.7.3 Topology planning

The first step is to define the topology of your cluster nodes and the networks that connect them. The node names are assigned in the panel shown in Figure 4 on page 13.


Cluster Planning Worksheets

Name: american
Type: HACMP/ES

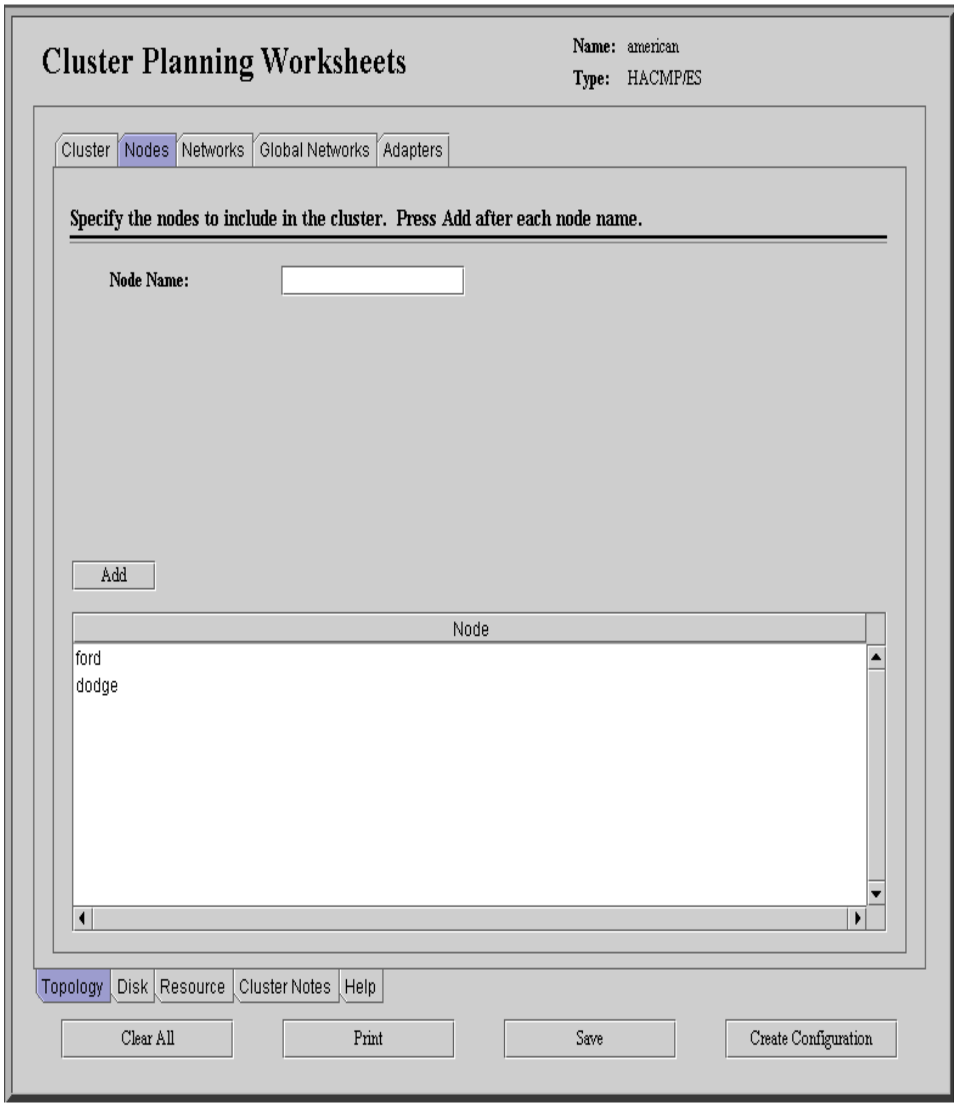
Specify the nodes to include in the cluster. Press Add after each node name.

Node Name:

Node
ford
dodge

Figure 4. Online worksheet: Nodes panel

Then, you define the networks as shown in Figure 5 on page 14.



Name: american
Type: HACMP/ES

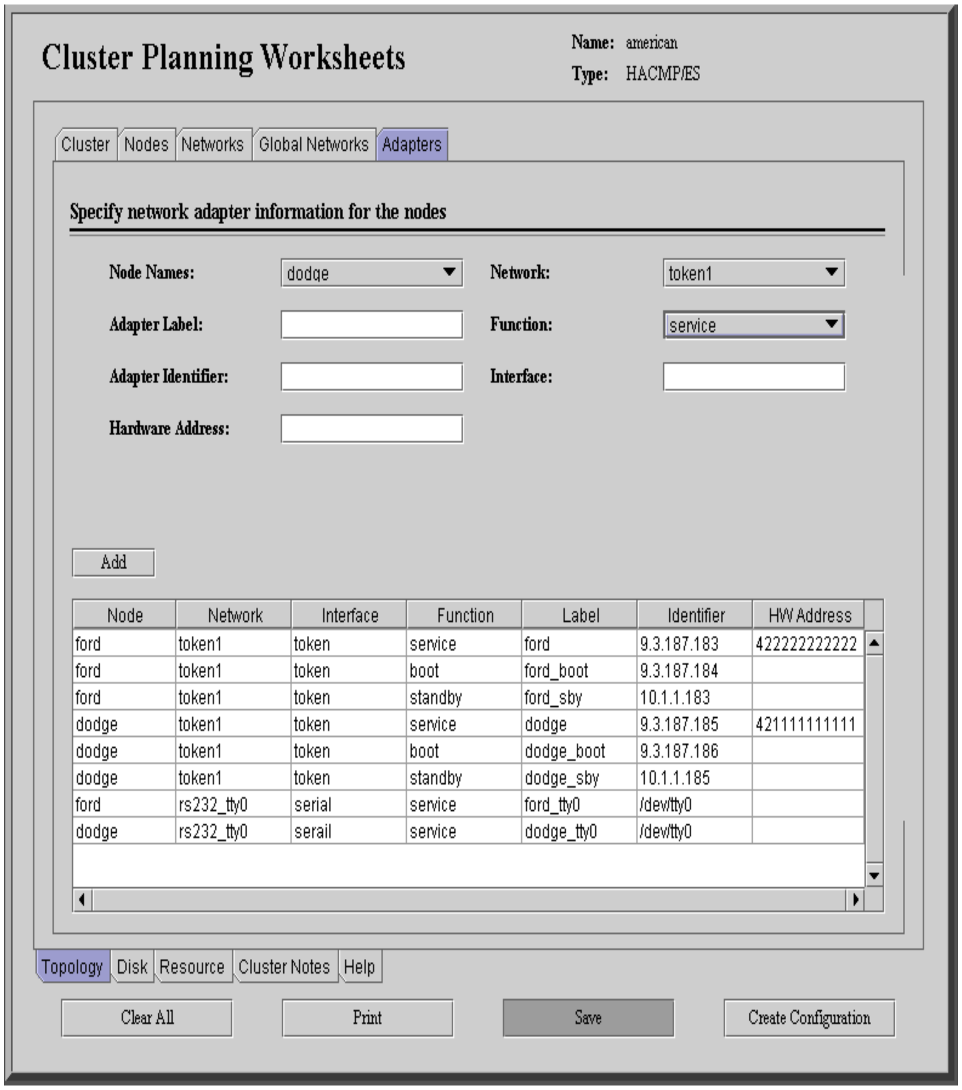
Specify the nodes to include in the cluster. Press Add after each node name.

Node Name:

Node
ford
dodge

Figure 5. Online worksheet: Network panel

Eventually, you enter the adapters as shown in Figure 6 on page 15.



Cluster Planning Worksheets

Name: american
Type: HACMP/ES

Specify network adapter information for the nodes

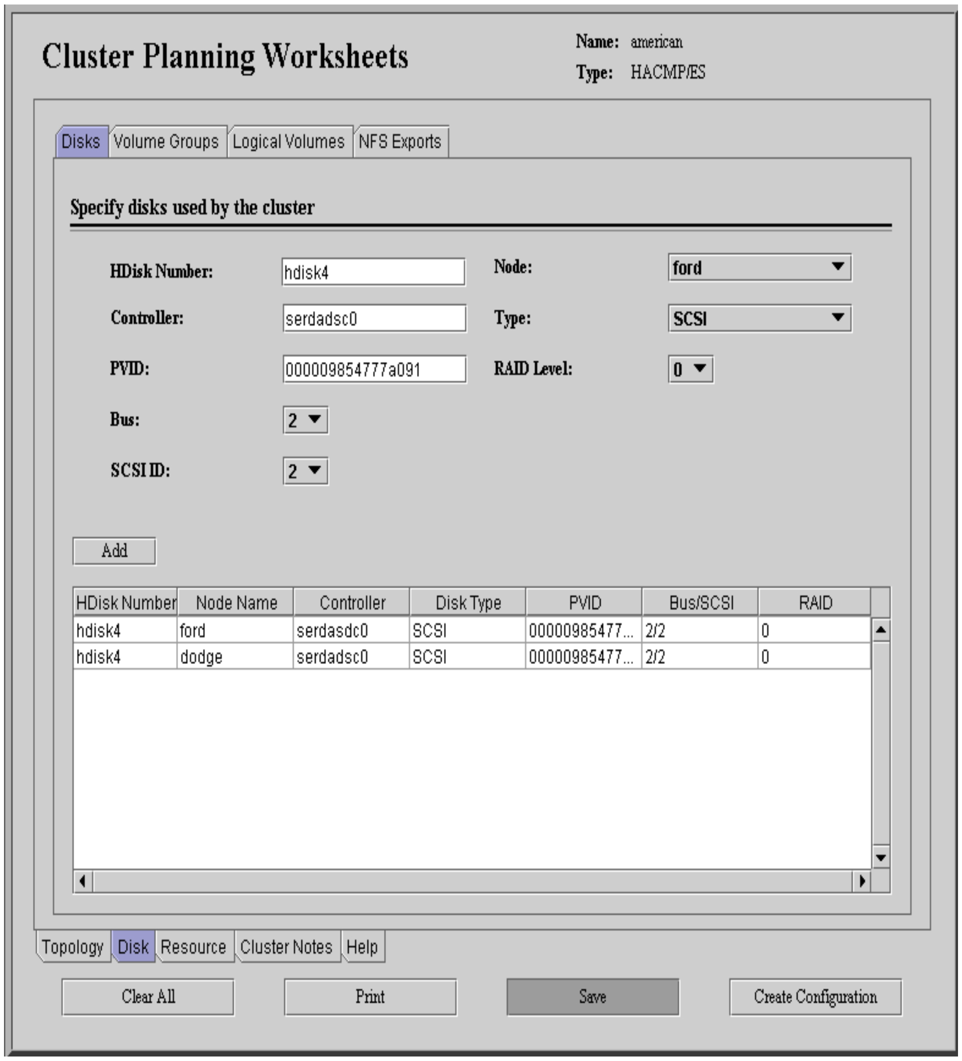
Node Names:	dodge	**Network:**	token1
Adapter Label:		**Function:**	service
Adapter Identifier:		**Interface:**	
Hardware Address:			

Node	Network	Interface	Function	Label	Identifier	HW Address
ford	token1	token	service	ford	9.3.187.183	422222222222
ford	token1	token	boot	ford_boot	9.3.187.184	
ford	token1	token	standby	ford_sby	10.1.1.183	
dodge	token1	token	service	dodge	9.3.187.185	421111111111
dodge	token1	token	boot	dodge_boot	9.3.187.186	
dodge	token1	token	standby	dodge_sby	10.1.1.185	
ford	rs232_tty0	serial	service	ford_tty0	/dev/tty0	
dodge	rs232_tty0	serail	service	dodge_tty0	/dev/tty0	

Figure 6. Online worksheet: Adapter panel

1.7.4 Disk planning

You can establish the shared disk configuration by selecting the **Disk** tab. In the first panel as shown in Figure 7 on page 16, you insert the disk details.



Cluster Planning Worksheets

Name: american
Type: HACMP/ES

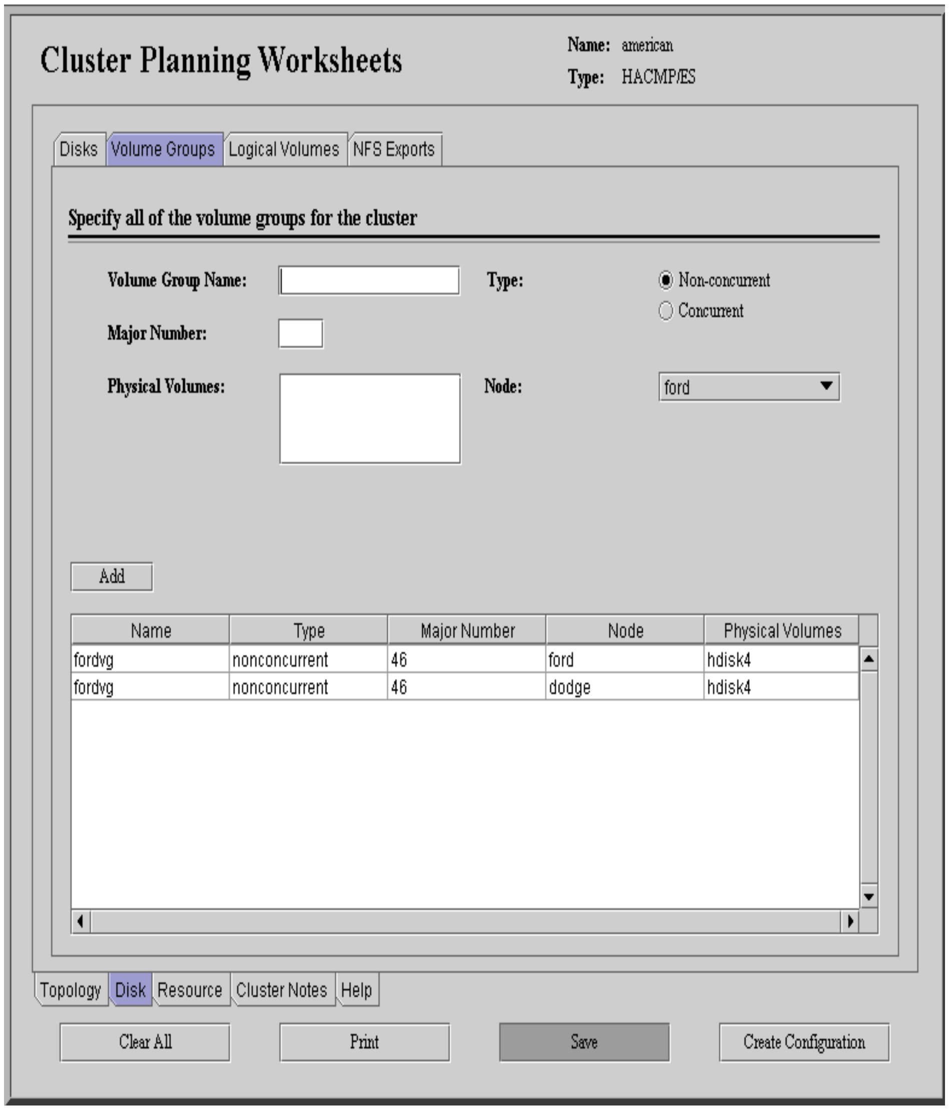
Specify disks used by the cluster

HDisk Number:	hdisk4	Node:	ford
Controller:	serdadsc0	Type:	SCSI
PVID:	000009854777a091	RAID Level:	0
Bus:	2		
SCSI ID:	2		

HDisk Number	Node Name	Controller	Disk Type	PVID	Bus/SCSI	RAID
hdisk4	ford	serdasdc0	SCSI	00000985477...	2/2	0
hdisk4	dodge	serdadsc0	SCSI	00000985477...	2/2	0

Figure 7. Online worksheet: Disk panel

Then, you select the **Volume Groups** tab to define the shared volume groups as shown in Figure 8 on page 17.

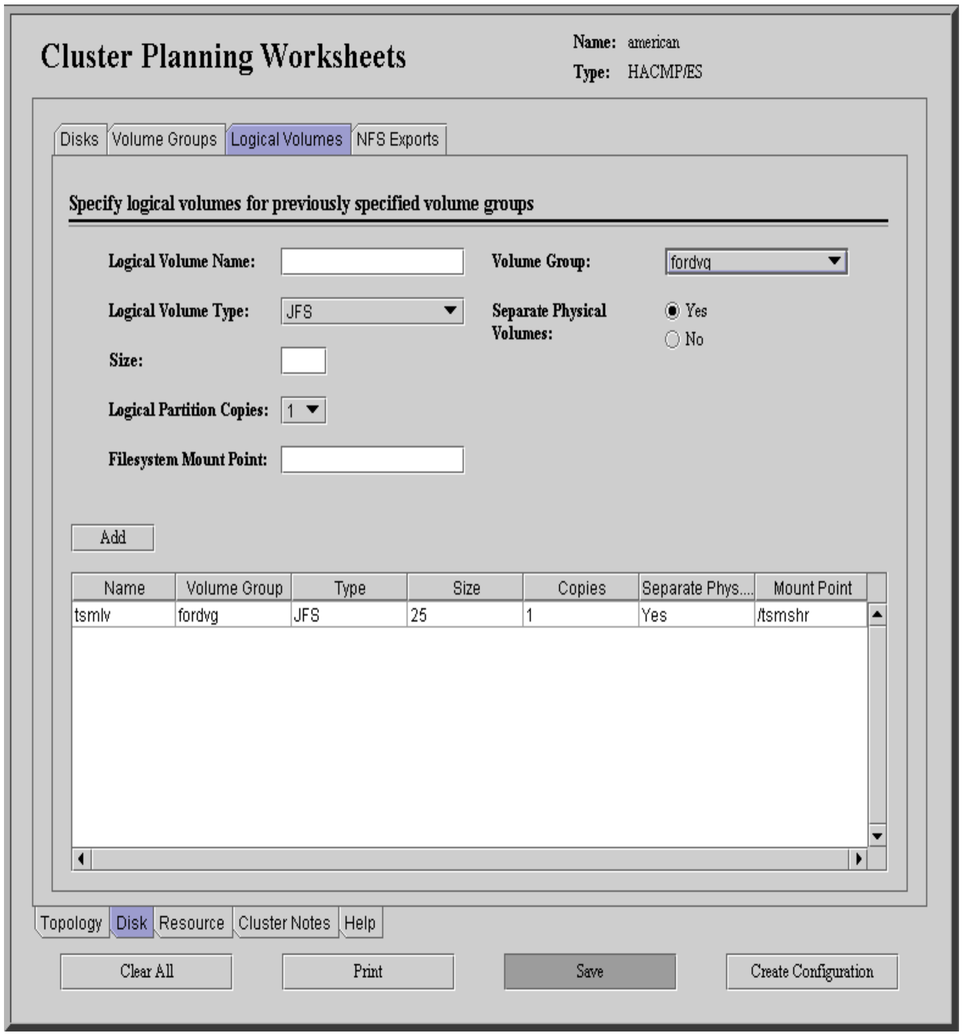


Type: [✓ Non-concurrent] Concurrent

Name	Type	Major Number	Node	Physical Volumes
fordvg	nonconcurrent	46	ford	hdisk4
fordvg	nonconcurrent	46	dodge	hdisk4

Figure 8. Online worksheet: Volume Group panel

Next, you list the shared logical volumes as shown in Figure 9 on page 18.


Cluster Planning Worksheets

Name: american
Type: HACMP/ES

Specify logical volumes for previously specified volume groups

Logical Volume Name:		Volume Group:	fordvg
Logical Volume Type:	JFS	Separate Physical Volumes:	[✓ Yes] No
Size:			
Logical Partition Copies:	1		
Filesystem Mount Point:			

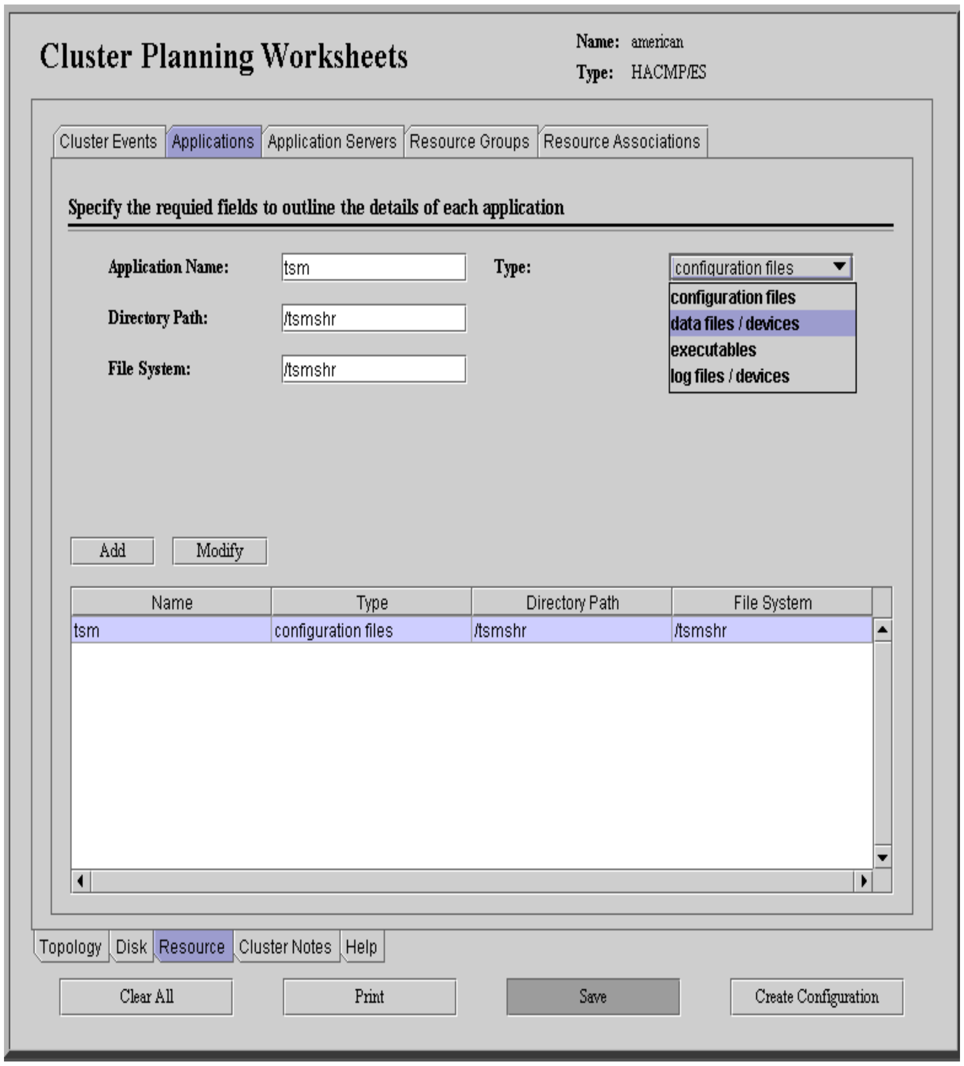
Name	Volume Group	Type	Size	Copies	Separate Phys....	Mount Point
tsmlv	fordvg	JFS	25	1	Yes	/tsmshr

Figure 9. Online worksheet: Logical Volumes panel

Eventually, you add the NFS file to be exported.

1.7.5 Resource planning

The Resource tab at the bottom of the window enables the four top tabs. The Cluster Events tab accesses the screen to specify events and associated scripts. You type in the command or script path for each event. The Applications tab shows the screen to assign a name to each application and to enter its directory path and file system as shown in Figure 10 on page 19.



Name: american
Type: HACMP/ES

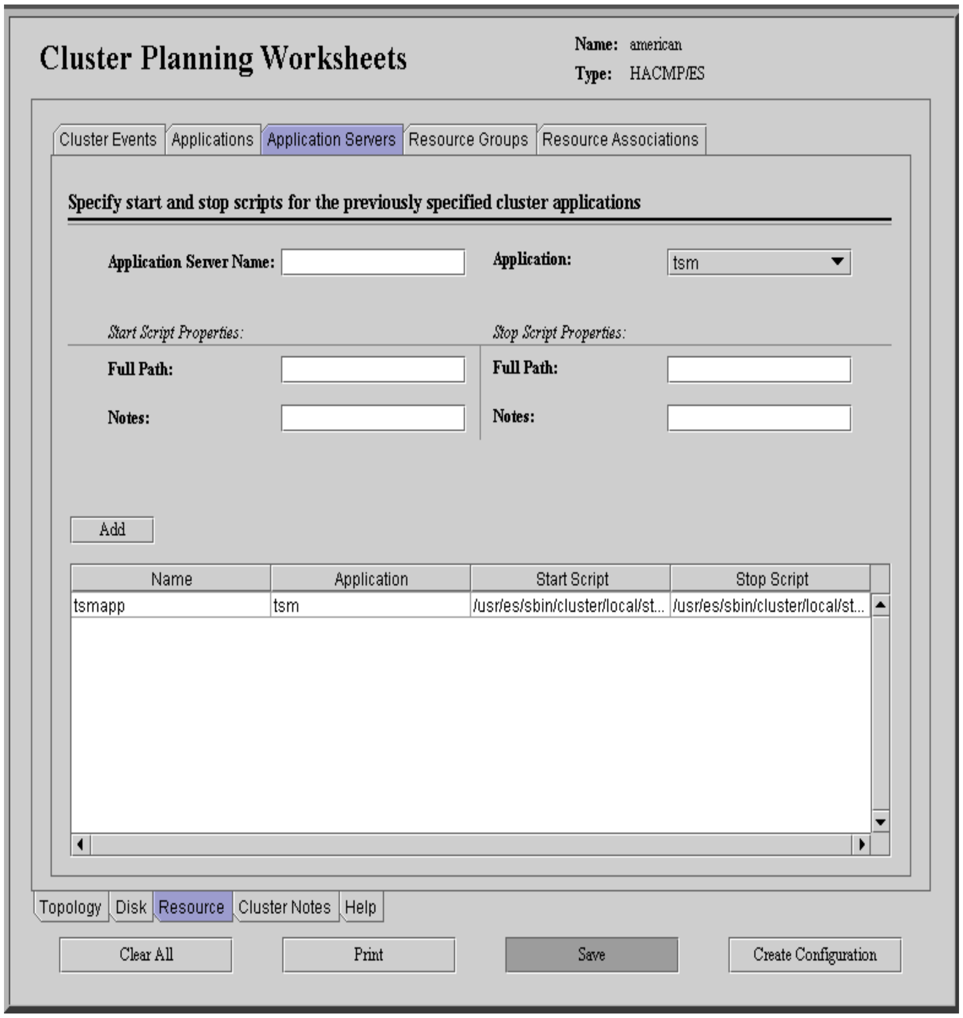
Specify the requied fields to outline the details of each application

Application Name: tsm
Directory Path: /tsmshr
File System: /tsmshr
Type: configuration files

Name	Type	Directory Path	File System
tsm	configuration files	/tsmshr	/tsmshr

Figure 10. Online worksheet: Applications panel

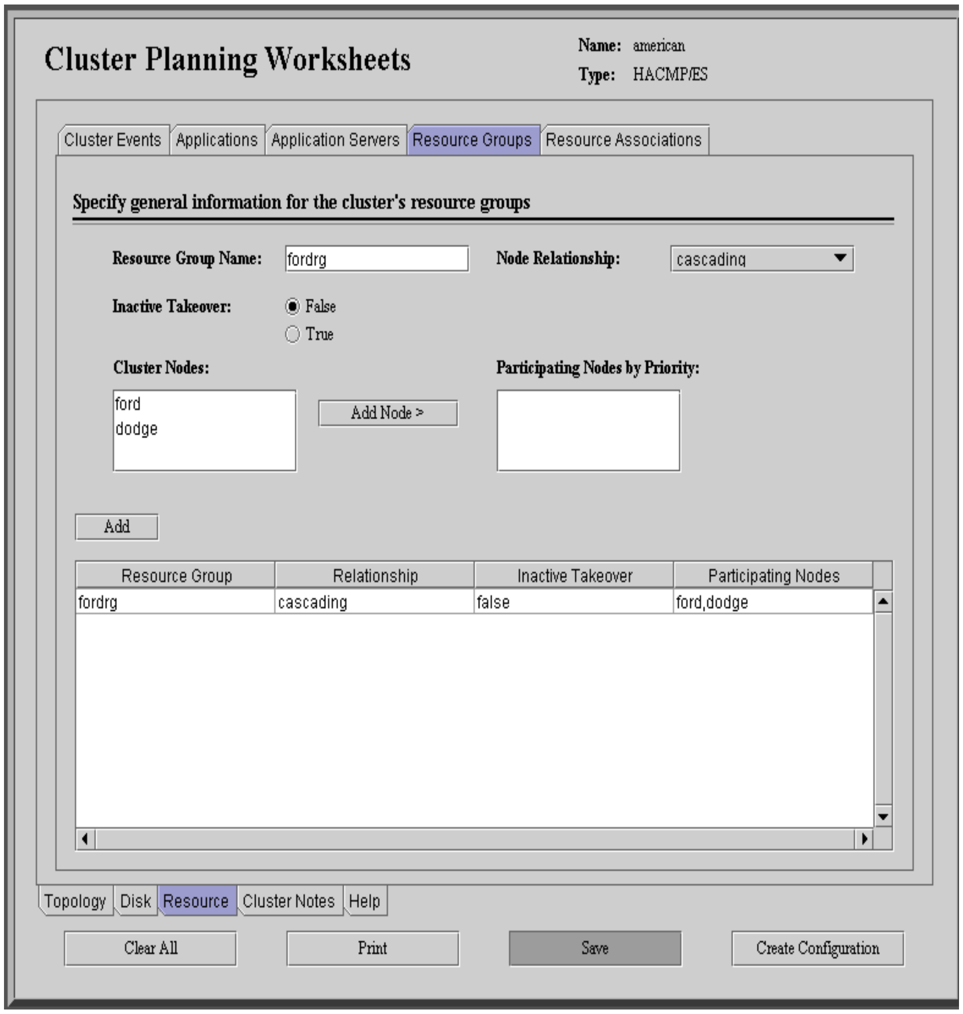
The next step is to select the **Applications Server** tab where you define the application server, the start script, and the stop script as shown in Figure 11 on page 20.



Name	Application	Start Script	Stop Script
tsmapp	tsm	/usr/es/sbin/cluster/local/st...	/usr/es/sbin/cluster/local/st...

Figure 11. Online worksheet: Applications Server panel

The Resource Groups tab lets you define the resource groups as shown in Figure 12 on page 21.



Specify general information for the cluster's resource groups

Resource Group Name: fordrg

Node Relationship: cascading

Inactive Takeover: (● False) ○ True

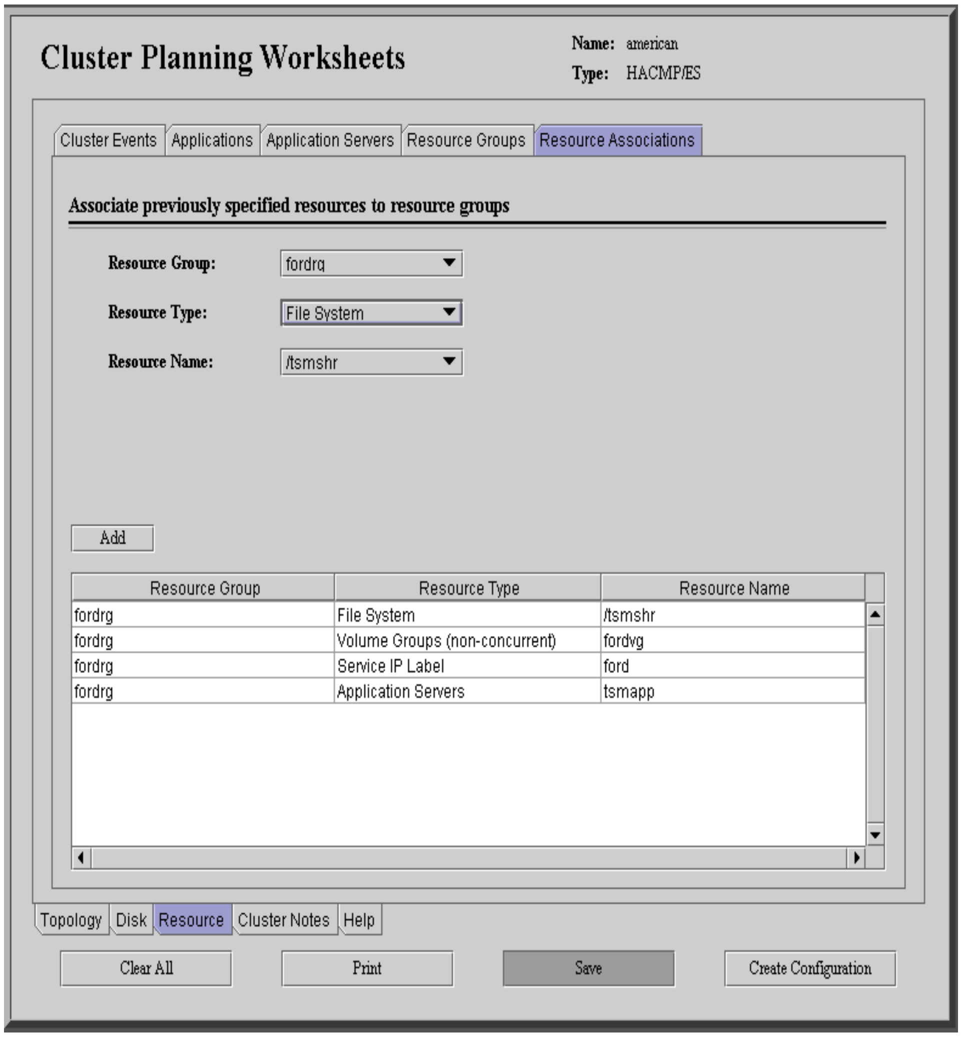
Cluster Nodes: ford, dodge

Participating Nodes by Priority:

Resource Group	Relationship	Inactive Takeover	Participating Nodes
fordrg	cascading	false	ford,dodge

Figure 12. Online worksheet: Resource Groups panel

Eventually, you associate the resources to the resource groups by selecting **Research Associations**, the right-most top tab, as shown in Figure 13 on page 22.



Resource Group: fordrg
Resource Type: File System
Resource Name: /tsmshr

Resource Group	Resource Type	Resource Name
fordrg	File System	/tsmshr
fordrg	Volume Groups (non-concurrent)	fordvg
fordrg	Service IP Label	ford
fordrg	Application Servers	tsmapp

Figure 13. Online worksheet: Resource Associations panel

1.7.6 Applying worksheet data to AIX cluster nodes

When the worksheets are ready, and you have entered all the cluster data, you can save the worksheet and create the configuration file (for example, cluster.conf) with the base panel buttons. Then, you transfer cluster.conf to one cluster node via FTP (it is a plain ASCII file), and you can use it to load the cluster configuration. This procedure does not supply a complete cluster building, but it aids in the cluster configuration. Therefore, the usual steps must be completed before applying the configuration file to the cluster, in particular:

- The network adapters must be configured.
- The LVM components must be defined via the usual AIX commands or exploiting the C-SPOC facilities.
- The files /etc/hosts, /.rhosts, and /usr/es/sbin/cluster/etc/clhosts must be edited and properly modified.

Note

If the new configuration is replacing an existing one, any current configuration must be saved in a snapshot.

Once you get the configuration file cluster.conf on your cluster node, complete the following two steps:

1. Stop the cluster services on all nodes.
2. On the cluster node, run the `cl_opsconfig` command:

   ```
   /usr/es/sbin/cluster/utilities/cl_opsconfig <your configuration file>
   ```

The cl_opsconfig utility automatically performs a synchronization, including verification, of the configuration. During this process, you see a series of messages indicating the events taking place and any warnings of errors.

Chapter 2. Configuration management in your cluster

HACMP/6000 clusters are used to replace single systems, inherently vulnerable to component failures, with groups of systems or clusters, where failing components can be backed up by redundant components. This presents the cluster as a highly available application server. In this environment, the cluster is thought of as the computer, rather than any particular machine. Since HACMP/6000 clusters are treated as a single entity, it is desirable to be able to manage the machines in the cluster as a single entity. In this chapter, we concentrate on managing the configurations of multiple machines in as integrated a fashion as possible.

2.1 Cluster configuration challenges

There are several areas of configuration management on clusters that are discussed in this and following chapters. A description of major areas follows in the next sections.

2.1.1 Software installation

HACMP/6000 does not directly affect software installation. As a general rule, you are encouraged to install any software products to be used in the cluster onto the internal disks of each of the machines. There may, however, be some cases where you do need to install software on the shared disks in the cluster. If this is done, you should carefully watch to see if the software installation automatically increases the size of any shared file systems (as might be done automatically by installp). If the installation process affects the size of a shared file system, you have to ensure that the change is known to all the other nodes in the cluster that may be acquiring the shared file system.

In an environment where you have many machines running the same software, it is a good idea to use an installation server. This makes installation faster and easier. It also provides a common place in the network where your installation images are kept.

Tivoli Software Distribution can be useful in an environment where there is a cluster of machines. With this product, you can manage software installation on a number of machines from a central site.

Section 2.3 on page 30 explains how you apply fixes in a running cluster to allow software maintenance in a non-stop environment.

2.1.2 Shared disk space

Managing shared disk space is a special area of administrative concern for HACMP/6000 clusters. Changes made to shared disk structures, such as volume groups, file systems, and logical volumes on one system, must also be communicated to other systems in the cluster. This is done by using the practice of exporting and reimporting the affected volume groups from the systems other than where the change was made.

In Section 2.4 on page 31, we explore the C-SPOC utility to simplify the management of shared disks.

2.1.3 Security and users

In HACMP/6000 clusters, it is typical that the same users are active on multiple machines, depending on the state of the cluster nodes and other components. It is desirable to keep this as transparent as possible to them. One important job is to make sure that users in the cluster have the same passwords, file and executable permissions, home directories, and shell environments on any of the cluster nodes on which they may be working.

HACMP/6000 does not automatically synchronize user information, so extra effort is often necessary in setting up a cluster. We discuss this subject more in Chapter 3, “Cluster user IDs and user environments” on page 49. In short, there are three possible tools to help us manage and synchronize user information:

- Network Information Service (NIS)
- C-SPOC
- The `rdist` command

2.2 Good practice in cluster management

The main objective of a proper cluster management is to maximize systems availability. A poorly managed cluster produces significant planned and unplanned down time. The following sections describe good practices for managing an HACMP cluster.

2.2.1 Plan for your cluster

Good cluster management should start with proper capacity planning before cluster implementation. You should identify the systems and user requirements to avoid changes to the cluster environment that result in system down time. The following is information that should be considered in HACMP planning:

- Storage requirement: Storage is probably the fastest growing resource in a system. You should be able to predict the disk storage requirements for at least a year so that you can select the best storage device for your cluster.
  - Buy a disk enclosure that allows you to hot-plug a disk.
  - Make sure there is enough free slots for disk upgrade. In case you need an upgrade, you can plug in the disk and configure it on the fly.
  - The disk cabling should allow the addition of new disks without having to recable.
- Network requirement: The network that you choose should be able to handle the network bandwidth of your environment. You cannot upgrade a network without causing down time. If possible, provide multiple paths from users to the cluster so that a network component failure does not result in system unavailability.

Remember that planning is the key to successfully administering HACMP for an AIX system because you must consider the effects of changes on the highly available applications and their use.

2.2.2 Assign an administrator for cluster maintenance

The HACMP for AIX system provides a highly available environment that ensures that critical applications at your site are available to end users. As system administrator, your job is to make sure that HACMP for AIX itself is stable and operational.

Cluster maintenance can be very sensitive, and it should be assigned to a dedicated person who knows the setup and the working environment of a cluster. Avoid having multiple administrators making changes to the cluster. Any change to the cluster should be properly planned and tested for possible outcomes.

As the system administrator of a highly available environment, you should strive to minimize cluster down time. Whenever possible, use the HACMP for AIX *dynamic reconfiguration* capability to change an active cluster without requiring down time. For those changes that require the cluster be shut down, schedule the down time for off hours when there is minimal activity on the system. When you plan to take the system down, notify end users in advance and inform them when services are restored.

2.2.3 Verify the cluster topology and resources

If you change your cluster topology or the configuration of a cluster node, you should perform the cluster verification procedure to make sure that all nodes agree on the cluster topology, network configuration, and ownership of resources. Use the `/usr/sbin/cluster/diag/clverify` command to verify a cluster configuration.

2.2.4 Backup your system

The practice of allocating multiple copies of a logical volume can enhance high availability in a cluster environment, but it should not be considered a replacement for regular system backups. Although HACMP for AIX is designed to survive failures within the cluster, it cannot survive a catastrophic failure where multiple points of failure leave data on disks unavailable. Therefore, it is imperative that you back up your system on a regular basis. You must have a backup procedure in place to ensure data reliability and to protect against catastrophic physical volume failure.

To maintain your HACMP for AIX environment, you must back up the root volume group (which contains the HACMP for AIX software) and the shared volume groups (which contain the data for highly available applications) regularly. HACMP for AIX is like other AIX environments from this perspective. Back up all nodes.

2.2.5 Document your system

As your HACMP for AIX system grows and changes, it differs from its initial cluster configuration. It is your responsibility as system administrator to document all aspects of the HACMP for AIX system unique to your environment. This responsibility includes documenting procedures concerning the highly available applications, recording changes that you make to the configuration scripts distributed with HACMP for AIX, documenting any custom scripts you write, recording the status of backups, maintaining a log of user problems, and maintaining records of all hardware.

Administration tasks on a cluster can be easier and safer if they are performed according to the written procedures. A proper document allows a smooth transition between administrators. A good document saves recovery time in an emergency situation.

After you configure the topology and resources of a cluster, you can use the cluster snapshot utility to save, in a file, the ODM information that defines the cluster. The saved configuration can later be used to restore the configuration. A cluster snapshot can also be applied to an active cluster to dynamically reconfigure the cluster.

2.2.6 Use C-SPOC whenever possible

The C-SPOC utility allows you to perform some administrative tasks on all cluster nodes by executing distributive command scripts that execute across the cluster. C-SPOC helps a system administrator to manage an HACMP cluster error free. Changes made through C-SPOC do not require cluster down time.

Use the conventional methods to perform tasks that are not supported by C-SPOC. Refer to Chapter 1, “New functions in HACMP/ES 4.3.1” on page 1 for more information about features provided by C-SPOC.

C-SPOC may be slower than conventional methods, but it provides maximum system availability.

2.2.7 Monitor your cluster

By design, HACMP for AIX masks various failures that occur within a cluster from end users. For example, HACMP for AIX masks a network adapter failure by swapping in a standby adapter. As a result, it is possible that a component in the cluster can fail and that you can be unaware that a failure has occurred. The danger here is that, while HACMP for AIX can survive one or possibly several failures, an unnoticed failure threatens a cluster’s ability to handle future failures.

To avoid this situation, it is recommended that you customize your HACMP for AIX system by adding event notification to the scripts designated to handle the various cluster events. You can also use the AIX Error Notification facility to provide warnings about hardware errors that do not cause HACMP for AIX events.

Error notification is provided with HACMP to help a system administrator manage the cluster proactively. Error notification is a useful tool. It helps you fix a problem before it becomes critical, therefore, giving you more room to schedule down time for maintenance if necessary. For more information about error notification, please refer to Chapter 13, “AIX Error Notification and device monitoring” on page 205.

Moreover, as a general practice, you should periodically inspect your cluster to make sure that it works properly. Schedule a time to perform a “fire drill” on the cluster. Test that the failover works accordingly.

2.2.8 Separate a production environment for a test environment

If possible, separate a production cluster from the test cluster. Freeze all changes on the production system until they successfully test on the test environment.

2.3 Applying software maintenance to an HACMP cluster

You can install software maintenance, called Program Temporary Fixes (PTFs), to your HACMP cluster while running HACMP for AIX cluster services on cluster nodes; however, you must stop cluster services on the node on which you are applying a PTF.

To apply software maintenance to your HACMP cluster:

1. Use the `smit clstop` fastpath to stop the cluster services on the node on which the PTF is to be applied. If you would like the resources provided by this node to remain available to users, stop cluster with takeover so that the takeover node continues to provide these resources to users.
2. Apply the software maintenance to this node using the procedure described in the *AIX V4.3 Installation Guide*, SC23-4112 and in any documentation distributed with the PTF.
3. Run the `/usr/sbin/cluster/diag/clverify` utility to ensure that no errors exist after installing the PTF.

Note

If cluster services are active on any cluster node and service adapters are configured with their service versus boot addresses (IPAT is enabled), a message appears indicating that the node is not on its boot address. This message does not reflect a problem with the cluster configuration; instead, it indicates that cluster services are running on the specified node.

4. Reboot the node to reload any HACMP for AIX kernel extensions that may have changed as a result of the PTF being applied. If an update to the cluster.base.client.lib file set has been applied, and you are using Cluster Lock Manager or Clinfo API functions, you may need to re-link your applications.
5. Restart the HACMP for AIX software on the node using the `smit clstart` fastpath and verify that the node successfully joined the cluster.
6. Repeat Steps 1 through 5 on the remaining cluster nodes.

2.4 Using C-SPOC to manage an HACMP cluster

Cluster-Single Point of Control (C-SPOC) is a utility that allows you to perform command maintenance tasks on all nodes in a cluster by executing a C-SPOC command on any node. The C-SPOC utility provides commands to manage cluster services, shared LVM, and group accounts.

C-SPOC enables a system administrator to manage an HACMP cluster easily and efficiently, therefore, reducing the planned down time significantly.

2.4.1 Managing cluster services

Starting and stopping cluster services may not be the most frequent tasks performed by a cluster administrators, but they are very important. C-SPOC allows a cluster administrator to start and stop a cluster from a single node in a cluster. Instead of starting up cluster services on individual nodes in a cluster, you can start the cluster services centrally from any of the cluster nodes. If you have a large cluster, you will appreciate the C-SPOC facility.

2.4.1.1 Starting cluster services

To start cluster services, use C-SPOC as shown in Figure 14.

```
ford::root:/>smit cl_admin
     HACMP for AIX Cluster Services
          Start Cluster Services
```

Figure 14. Starting cluster services using C-SPOC

The screen shown in Figure 15 on page 32 appears.

```
                    Start Cluster Services

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Start now, on system restart or both                now +
  Start Cluster Services on these nodes              [dodge,ford] +
  BROADCAST message at startup?                       false +
  Startup Cluster Lock Services?                      false +
  Startup Cluster Information Daemon?                 false +


F1=Help            F2=Refresh         F3=Cancel          F4=List
F5=Reset           F6=Command         F7=Edit            F8=Image
F9=Shell           F10=Exit           Enter=Do
```

Figure 15. Options for starting cluster services

At the Start Cluster Services on these nodes field, press **F4** to select from a list. By pressing **F7**, select all the cluster nodes that you want to start the cluster services. Set the Startup Cluster Information Daemon? field to `true`. Press **Enter** to start the cluster services.

The cluster services are started on all the nodes you have selected. You can monitor the progress of the cluster startup by monitoring the hacmp log file on each node as follows:

```
# tail -f /tmp/hacmp.out
```

Note

When starting cluster services, it is always recommended to start the Cluster Information Daemon (clinfo) along with the Cluster Manager. Clinfo is a daemon that monitors the cluster status. With clinfo running, you can use the clstat utility to track the cluster status. Clstat uses clinfo to get the cluster status.

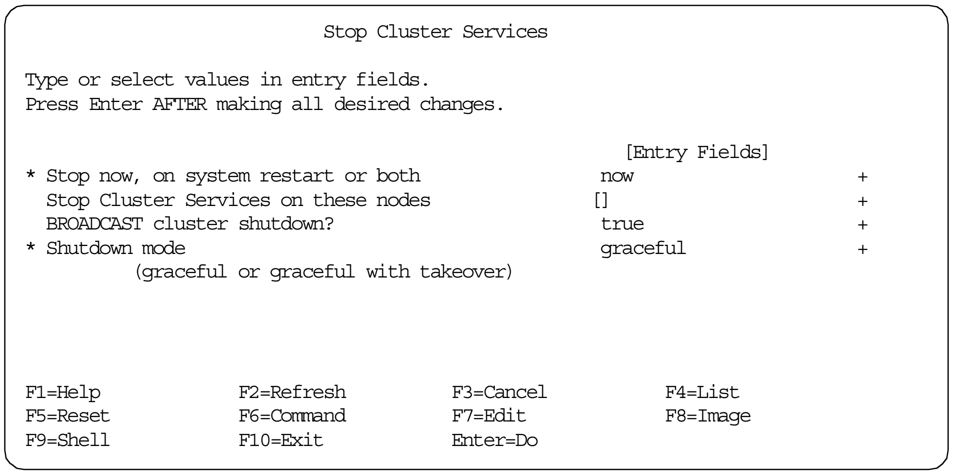
2.4.1.2 Stopping cluster services

To stop cluster services, use C-SPOC as shown in Figure 16.

```
ford::root:/>smit cl_admin
     HACMP for AIX Cluster Services
          Stop Cluster Services
```

Figure 16. Stopping cluster services using C-SPOC

The screen shown in Figure 17 appears.



```
                              Stop Cluster Services

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                         [Entry Fields]
* Stop now, on system restart or both                  now                   +
  Stop Cluster Services on these nodes                []                    +
  BROADCAST cluster shutdown?                          true                  +
* Shutdown mode                                        graceful              +
          (graceful or graceful with takeover)




F1=Help              F2=Refresh           F3=Cancel            F4=List
F5=Reset             F6=Command           F7=Edit              F8=Image
F9=Shell             F10=Exit             Enter=Do
```

Figure 17. Options for stopping cluster services

At the Stop Cluster Services field, press **F4** to select from a list. By pressing **F7**, select all the cluster nodes that you want to stop the cluster services. Select the Shutdown mode from either **graceful**, or **graceful with takeover**. Press **Enter** to initiate the cluster shutdown.

The cluster services are stopped on all the nodes you have selected. You can monitor the progress of the cluster shutdown by monitoring the HACMP log file on each node as follows:

```
# tail -f /tmp/hacmp.out
```

2.4.2 Managing shared LVM

C-SPOC provides an easy way to perform changes in the HACMP shared LVM configurations. The following are tasks that can be performed using C-SPOC:

- Shared volume group management
- Shared logical volume management
- Shared file systems management

In order to understand how C-SPOC can help in managing shared LVM, we look at several scenarios of how we would manage the changes in order to minimize down time.

All scenarios presented in this section refer to the cluster diagram shown in Figure 18.

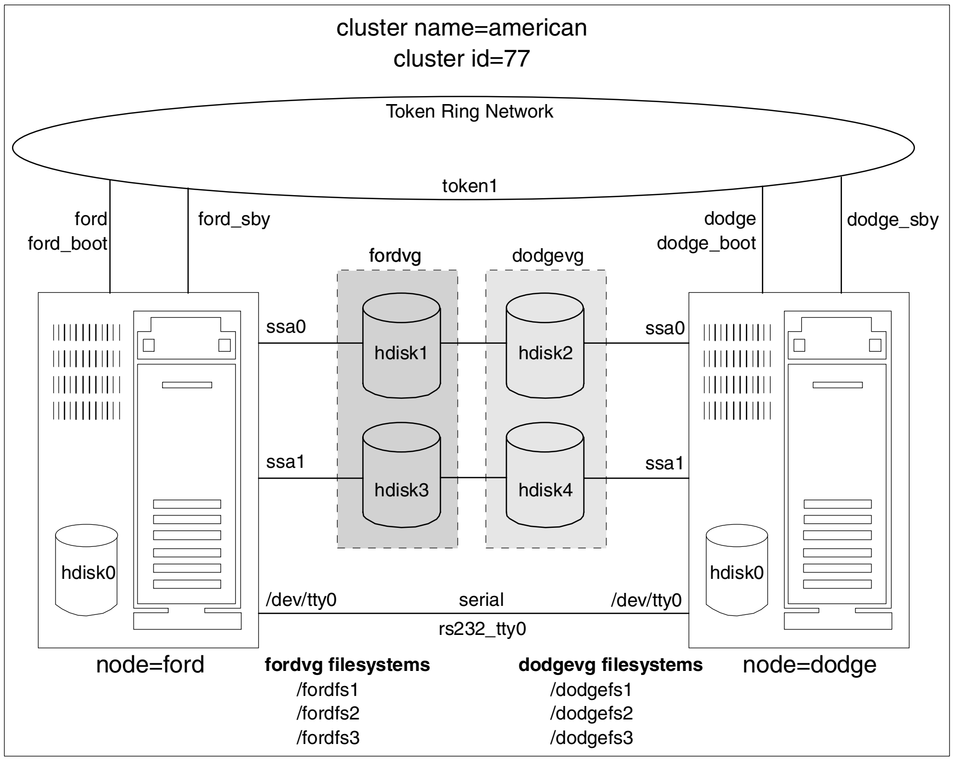


Figure 18. Cluster diagram

2.4.2.1 Extending the size of the shared JFS

To extend the /fordfs1 file system, perform the steps shown in Figure 19.

```
ford::root:/>smit cl_admin
     Cluster Logical Volume Manager
          Shared File Systems
               Change / Show Characteristics of a Shared File System
```

Figure 19. Extending shared JFS using C-SPOC

You get a panel as shown in Figure 20 on page 35.

```
 File System Name

 Move cursor to desired item and press Enter.

     fordrg  /fordfs1
     fordrg  /fordfs2
     fordrg  /fordfs3

 F1=Help                 F2=Refresh              F3=Cancel
 F8=Image                F10=Exit                Enter=Do
 /=Find                  n=Find Next
```

Figure 20. Selecting a file system to extend

Select `/fordfs1` from the list.

```
 Change / Show Characteristics of a Shared File System in the Cluster

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
  Resource Group Name                                fordrg
  File system name                                   /fordfs1
  NEW mount point                                   [/fordfs1]
  SIZE of file system (in 512-byte blocks)          [2000000]
  Mount GROUP                                       []
  Mount AUTOMATICALLY at system restart?             no                   +
  PERMISSIONS                                        read/write           +
  Mount OPTIONS                                     []                    +
  Start Disk Accounting?                             no                   +
  Fragment Size (bytes)                              4096
  Number of bytes per inode                          4096
  Compression algorithm                              no




F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 21. Changing the file system size

Fill in the required value in the SIZE field of file system. Press **Enter** to commit your changes. Once completed, the information that you have just changed is be synchronized to all the cluster nodes. Press **F10** to exit.

You can use the same procedure to change any other characteristic of a shared JFS.

2.4.2.2 Adding a shared JFS

Adding a shared JFS using C-SPOC is similar to adding file systems in the previously defined logical volume in the regular LVM management. The process of creating a shared file system involves four steps:

1. Create a shared logical volume in the LVM.
2. Create a shared JFS on the logical volume created.
3. Add a shared JFS in the resource group.
4. Synchronize the cluster resources.

C-SPOC does not support creation of JFS directly into a shared volume group, allowing some control on the file system characteristics.

In this scenario, we add a file system, /fordfs4, in fordvg with the size of 64 MB.

First, we create a shared logical volume as shown in Figure 22.

```
ford::root:/>smit cl_admin
     Cluster Logical Volume Manager
          Shared Logical Volumes
               Add a Shared Logical Volumem
```

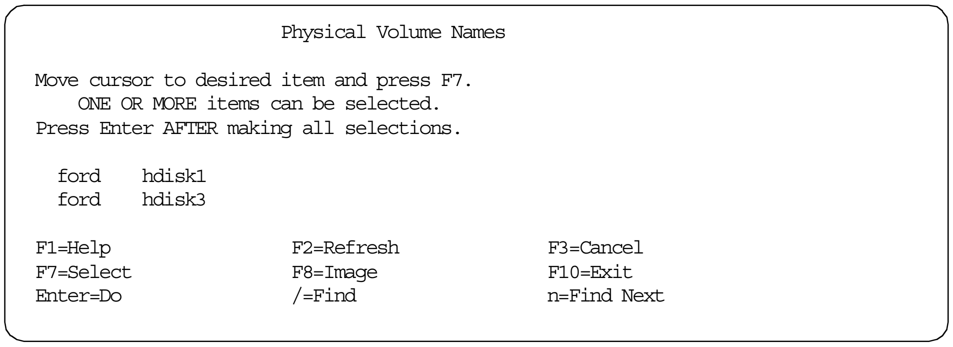
Figure 22. Adding a shared logical volume using C-SPOC

The panel shown in Figure 23 appears.

```
                         Shared Volume Group Names

 Move cursor to desired item and press Enter.

   fordrg fordvg
   dodgerg dodgevg

 F1=Help                  F2=Refresh               F3=Cancel
 F8=Image                 F10=Exit                 Enter=Do
 /=Find                   n=Find Next
```

Figure 23. Selecting the shared volume group to place the logical volume

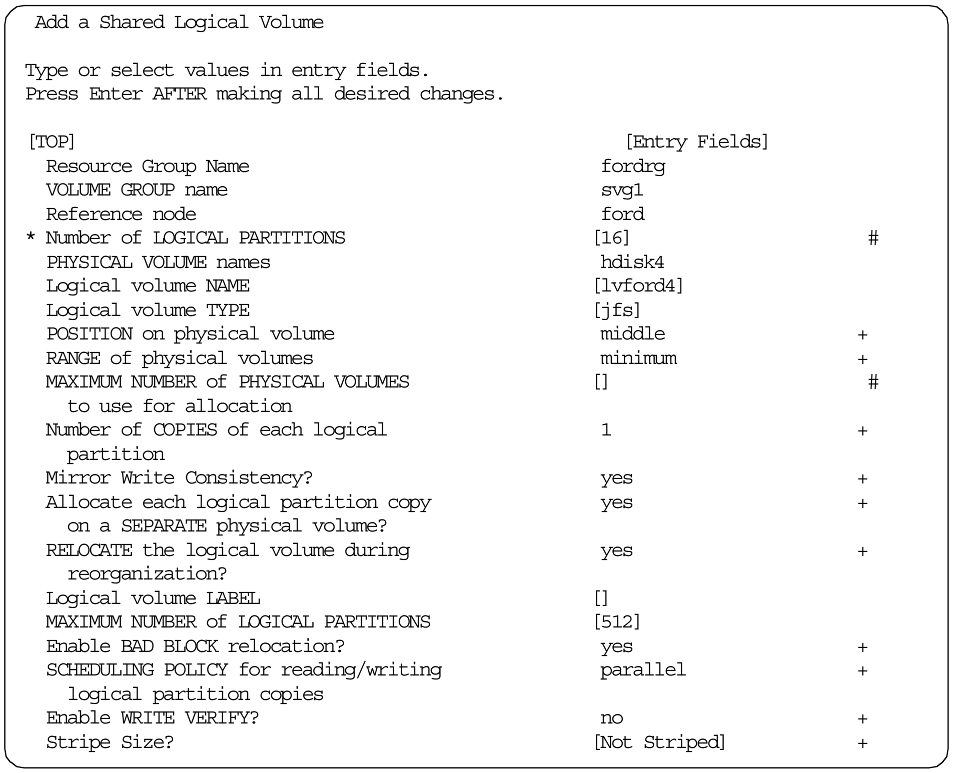
Select `fordvg`, and the panel shown in Figure 24 on page 37 appears.



```
                        Physical Volume Names

 Move cursor to desired item and press F7.
     ONE OR MORE items can be selected.
 Press Enter AFTER making all selections.

   ford    hdisk1
   ford    hdisk3

 F1=Help                F2=Refresh              F3=Cancel
 F7=Select              F8=Image                F10=Exit
 Enter=Do               /=Find                  n=Find Next
```

Figure 24. Selecting physical disk(s) to place the logical volume

Select `hdisk1` as the physical disk where the logical volume is created, and press **Enter** to commit the selection. The shared logical volume characteristics appear as shown in Figure 25.



```
 Add a Shared Logical Volume

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                    [Entry Fields]
  Resource Group Name                                  fordrg
  VOLUME GROUP name                                    svg1
  Reference node                                       ford
* Number of LOGICAL PARTITIONS                        [16]                    #
  PHYSICAL VOLUME names                                hdisk4
  Logical volume NAME                                 [lvford4]
  Logical volume TYPE                                 [jfs]
  POSITION on physical volume                          middle               +
  RANGE of physical volumes                            minimum              +
  MAXIMUM NUMBER of PHYSICAL VOLUMES                  []                      #
    to use for allocation
  Number of COPIES of each logical                     1                    +
    partition
  Mirror Write Consistency?                            yes                  +
  Allocate each logical partition copy                 yes                  +
    on a SEPARATE physical volume?
  RELOCATE the logical volume during                   yes                  +
    reorganization?
  Logical volume LABEL                                []
  MAXIMUM NUMBER of LOGICAL PARTITIONS                [512]
  Enable BAD BLOCK relocation?                         yes                  +
  SCHEDULING POLICY for reading/writing                parallel             +
    logical partition copies
  Enable WRITE VERIFY?                                 no                   +
  Stripe Size?                                        [Not Striped]         +
```

Figure 25. Shared logical volume characteristics

Fill in the number of logical partitions (LP) in the Number of LOGICAL PARTITIONS field. In our scenario, we use 16 LPs to represent 64 MB.

Fill in the Logical Volume Name as `lvford4`, and Logical volume TYPE as `jfs`. Press **Enter** to commit the changes.

The logical volume, lvford4, is now created on both cluster nodes.

Once the shared logical volume is created, create a file system on the logical volume. See Figure 26.

```
ford::root:/>smit cl_admin
     Cluster Logical Volume Manager
          Shared File Systems
               Create a Shared File System
                    Add a Standard Journaled File System
```

Figure 26. Creating a shared JFS using C-SPOC

The panel shown in Figure 27 appears.

```
Logical Volume Names

Move cursor to desired item and press F7.
    ONE OR MORE items can be selected.
Press Enter AFTER making all selections.

  lvford4   ford,dodge

F1=Help                F2=Refresh             F3=Cancel
F7=Select              F8=Image               F10=Exit
Enter=Do               /=Find                 n=Find Next
```

Figure 27. Selecting a logical volume

Select `lvford4`. The screen shown in Figure 28 on page 39 appears.

```
 Create a Shared File System

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                       [Entry Fields]
  Node Names                                          ford,dodge
  LOGICAL VOLUME name                                 lvford4
* MOUNT POINT                                        [/fordfs4]
  PERMISSIONS                                         read/write              +
  Mount OPTIONS                                      []                       +
  Start Disk Accounting?                              no                      +
  Fragment Size (bytes)                               4096                    +
  Number of bytes per inode                           4096                    +
  Allocation Group Size (MBytes)                      8                       +


F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 28. Characteristics of the JFS

Fill in the MOUNT POINT field as `/fordfs4`, and press **Enter** to commit the selection. Once the command completes, the shared file system is ready.

The newly created file system is not recognized by the cluster until the file system is added into one of the resource groups. To modify a resource group, perform the steps shown in Figure 29.

```
ford::root:/>smit hacmp
   Cluster Configuration
      Cluster Resources
         Change/Show Resources for a Resource Group
```

Figure 29. Modifying an HACMP resource group

The panel shown in Figure 30 on page 40 appears.

```
Select a Resource Group

Move cursor to desired item and press Enter.

  dodgerg
  fordrg

F1=Help                 F2=Refresh              F3=Cancel
F8=Image                F10=Exit                Enter=Do
/=Find                  n=Find Next
```

Figure 30. Selecting a resource group to be modified

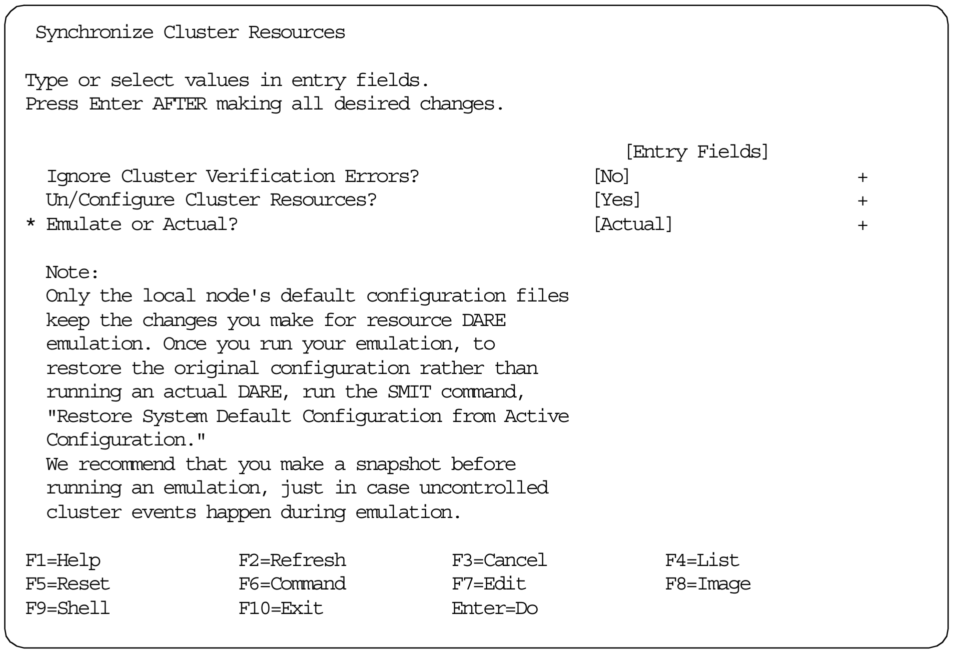
Select `fordrg` as the resource group, and the panel appears as shown in Figure 31.

```
                 Configure Resources for a Resource Group

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                      [Entry Fields]
  Resource Group Name                                    fordrg
  Node Relationship                                      cascading
  Participating Node Names                               ford dodge

  Service IP label                                      [ford]                +
  Filesystems                                           [rdfs2 /forfs3 /fordfs4] +
  Filesystems Consistency Check                          fsck                 +
  Filesystems Recovery Method                            sequential           +
  Filesystems to Export                                 []                    +
  Filesystems to NFS mount                              []                    +
  Volume Groups                                         [svg1]                +
  Concurrent Volume groups                              []                    +
  Raw Disk PVIDs                                        []                    +
  AIX Connections Services                              []                    +
[MORE...9]

F1=Help            F2=Refresh          F3=Cancel          F4=List
F5=Reset           F6=Command          F7=Edit            F8=Image
F9=Shell           F10=Exit            Enter=Do
```

Figure 31. Characteristics of the resource group

In the File systems field, add `/fordfs4` to the list of shared file systems, and press **Enter** to commit the changes.

The cluster resources need to be synchronized in order to propagate the cluster resource information to all cluster nodes. Press **F3** twice to get back to the Cluster Resources screen. Select **Synchronize Cluster Resources**, and the panel shown in Figure 32 on page 41 appears.



```
 Synchronize Cluster Resources

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                          [Entry Fields]
  Ignore Cluster Verification Errors?                  [No]                      +
  Un/Configure Cluster Resources?                      [Yes]                     +
* Emulate or Actual?                                   [Actual]                  +

  Note:
  Only the local node's default configuration files
  keep the changes you make for resource DARE
  emulation. Once you run your emulation, to
  restore the original configuration rather than
  running an actual DARE, run the SMIT command,
  "Restore System Default Configuration from Active
  Configuration."
  We recommend that you make a snapshot before
  running an emulation, just in case uncontrolled
  cluster events happen during emulation.

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 32. Cluster resources synchronization

Press **Enter** to start synchronizing the cluster resources. When the synchronization completes, the new file system is automatically mounted on node ford as part of Dynamic Reconfiguration (DARE) facility. A new shared file system is successfully added to the cluster.

Note

If you do not wish to mount the file system automatically after the synchronization process, set the field Un/Configure Cluster Resources to No.

2.4.2.3 Creating a shared volume group

Creating a shared volume group is defining unused disks into a volume group. To create a shared volume group, enter the information shown in Figure 33 on page 42.

```
ford::root:/>smit cl_admin
     Cluster Logical Volume Manager
          Shared Volume Groups
               Create a Shared Volume Group
```

Figure 33. Creating a shared volume group using C-SPOC

The panel shown in Figure 34 appears.

```
                                   Node Names

 Move cursor to desired item and press F7.
     ONE OR MORE items can be selected.
 Press Enter AFTER making all selections.

 > dodge
 > ford

 F1=Help                  F2=Refresh               F3=Cancel
 F7=Select                F8=Image                 F10=Exit
 Enter=Do                 /=Find                   n=Find Next
```

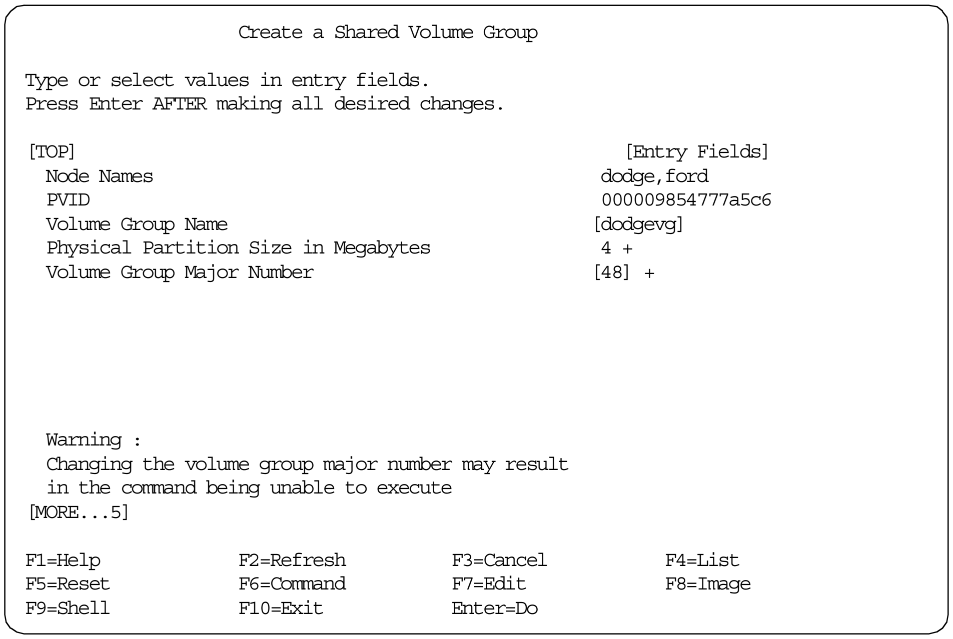
Figure 34. Selecting a node for the volume group

Use **F7** to select both nodes, and press **Enter** to commit the selection. The next screen as shown in Figure 35 appears.

```
                          Physical Volumes

 Move cursor to desired item and press F7.
     ONE OR MORE items can be selected.
 Press Enter AFTER making all selections.

 > 000009854777a5c6

 F1=Help                  F2=Refresh               F3=Cancel
 F7=Select                F8=Image                 F10=Exit
 Enter=Do                 /=Find                   n=Find Next
```

Figure 35. Selecting physical volumes for the volume group

Select the appropriate physical volume ID by pressing **F7**, and press **Enter**. The screen shown in Figure 36 on page 43 appears.



```
                    Create a Shared Volume Group

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                     [Entry Fields]
  Node Names                                            dodge,ford
  PVID                                                  000009854777a5c6
  Volume Group Name                                    [dodgevg]
  Physical Partition Size in Megabytes                  4 +
  Volume Group Major Number                            [48] +






  Warning :
  Changing the volume group major number may result
  in the command being unable to execute
[MORE...5]

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 36. Volume group characteristics

Fill in the Volume Group Name as `dodgevg`, and press **Enter** to commit the selection. The shared volume group is now successfully created.

Note

C-SPOC does not support management of a shared volume group until it is defined into the cluster resource group. If you want to use C-SPOC to create a file system or a logical volume in a new volume group, add the volume group name into the resource group immediately after you create it. Then, synchronize the resource group.

2.4.2.4 Changing a mirrored disk

Changing a mirrored disk in a cluster environment should be done with caution. In order to avoid unnecessary cleanup of the ODM after replacing a mirrored shared disk, proper steps have to be taken:

1. Remove the mirror copy from the failed disk.
2. Remove the physical volume from the volume group (reducevg).
3. Remove the disk definition from the system.
4. Replace the disk with a new disk.

5. Configure the new disk into the system.
6. Include the new disk in the same volume group.
7. Make a mirror copy on the new disk.
8. Synchronize the mirror.

Use C-SPOC to perform all steps with the exception of step four. Assuming that the shared disk is hot-pluggable, there is no down time incurred if you use C-SPOC.

2.5 Troubleshooting a dead man switch

The term “dead man switch” (DMS) describes the AIX kernel extension that causes a system panic and dump under certain cluster conditions if it is not reset. The dead man switch halts a node when it enters a hung state that extends beyond a certain time limit. This enables another node in the cluster to acquire the hung node’s resources in an orderly fashion, avoiding possible contention problems.

A problem is triggered when a node is experiencing an extreme performance problem, such as a large I/O transfer, excessive error logging, or running out of memory, and the Cluster Manager is starving for CPU time. Therefore, it cannot reset the dead man switch within the time allotted. Problem applications running at a priority higher than the Cluster Manager can also cause this problem because they can consume all available CPU time before the Cluster Manager gets any.

Solutions related to performance problems should be performed in the following order:

1. Tune the system using I/O pacing.
2. Increase the syncd frequency.
3. If needed, increase the amount of memory available for the communications subsystem.
4. Change the Failure Detection Rate.

2.5.1 Tuning the system using I/O pacing

Use I/O pacing to tune the system so that system resources are distributed more equitably during large disk writes. Enabling I/O pacing is required for an HACMP cluster to behave correctly during large disk writes, and it is strongly recommended if you anticipate large blocks of disk writes on your HACMP cluster.

You can enable I/O pacing using the `smit chgsys` fastpath to set high- and low-water marks. These marks are, by default, set to zero (disabling I/O pacing) when AIX is installed. While the most efficient high- and low-water marks vary from system to system, an initial high-water mark of 33 and a low-water mark of 24 provide a good starting point.

These settings only slightly reduce write times and consistently generate correct failover behavior from HACMP for AIX. If a process tries to write to a file at the high-water mark, it must wait until enough I/O operations have finished to make the low-water mark. See the *AIX Versions 3.2 and 4 Performance Monitoring and Tuning Guide*, SC23-2365 for more information about I/O pacing.

2.5.2 Increasing the syncd frequency

Edit the /sbin/rc.boot file to increase the syncd frequency from its default value of 60 seconds to either 30, 20, or 10 seconds. Increasing the frequency forces more frequent I/O flushes and reduces the likelihood of triggering the dead man switch due to heavy I/O traffic.

2.5.3 Increasing the memory for the communications subsystem

The memory for communication subsystems is represented by mbufs. Memory is not sufficient for communication to operate when output of netstat -m reports that requests for mbufs are being denied, or if errors indicating LOW_MBUFS are being logged to the AIX error report.

To increase required memory, you should increase the value associated with “thewall” network option. The default value is 25 percent of the real memory. This can be increased to as much as 50 percent of the real memory. To change this value, execute the following command:

```
no -o thewall= xxxxx
```

Where `xxxxx` is the value you want to be available for use by the communications subsystem. For example:

```
no -o thewall=131072
```

In order to make the change effective at system reboot, add the same command to the end of the /etc/rc.net file.

2.5.4 Changing the failure detection rate

By changing the failure detection rate of a network interface module (NIM), the HACMP cluster is tuned to be more tolerant with the heavy network traffic. You can change the failure detection rate by using the standard setting provided by SMIT, or you can modify the HACMPnim ODM object to suite your environment. Refer to the *HACMP V4.3 AIX: Troubleshooting Guide*, SC23-4280, for detailed information about how you can change the failure detection rate.

2.5.5 Changing the fibrillate value in topology services

HACMP/ES uses topology services to manage the cluster heartbeat. To determine which parameters are being used by HACMP/ES, use the command shown in Figure 37.

```
ford::root:/>lssrc -ls topsvcs
Subsystem         Group            PID     Status
 topsvcs          topsvcs          15680   active
 Network Name   Indx Defd Mbrs St Adapter ID      Group ID
 token1_0       [ 0]    2    2  S 9.3.187.183     9.3.187.185
 token1_0       [ 0]              0x409634a5      0x409634c3
 HB Interval = 1 secs. Sensitivity = 4 missed beats
 token1_1       [ 1]    2    2  S 10.1.1.183      10.1.1.185
 token1_1       [ 1]              0x40963423      0x4096345c
 HB Interval = 1 secs. Sensitivity = 4 missed beats
 rs232_0        [ 2]    2    2  S 255.255.0.1     255.255.0.1
 rs232_0        [ 2]              0x80963424      0x80963432
 HB Interval = 2 secs. Sensitivity = 4 missed beats
   2 locally connected Clients with PIDs:
 haemd(  9636) hagsd(  4098)
   Dead Man Switch Enabled:
      reset interval = 1 seconds
      trip  interval = 16 seconds
   Configuration Instance = 1
   Default: HB Interval = 1 secs. Sensitivity = 4 missed beats
```

Figure 37. Listing of topsvcs parameters

The command output shows that the two parameters that affect the cluster heartbeat rate are HB Interval and Sensitivity (Fibrillate Count).

HACMP/ES 4.3.1 uses the following formula to calculate time the taken to detect a node failure:

(heartbeat interval) * (fibrillate count) * 2 seconds

We suggest that you leave the HB interval at the default values and change the fibrillate count instead.

To change the fibrillate count, do the steps shown in Figure 38 on page 47.

```
ford::root:/>smit hacmp
     Cluster Configuration
          Cluster Topology
               Configure Topology Services and Group Services
                     Change / Show Topology and Group Services Configuration
```

Figure 38. Changing the fibrillate count using SMIT

The panel shown in Figure 39 then appears.

```
               Change / Show Topology and Group Services Configuration

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                          [Entry Fields]
* Interval between Heartbeats (seconds)              [1]                      #
* Fibrillate Count                                   [4]                      #
* Topology Services log length (lines)               [5000]                   #
* Group Services log length (lines)                  [5000]                   #



F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 39. Topology and group services configuration

Change the Fibrillate Count from 4 to 8, which gives a failure detection time of 16 seconds. Press **Enter** to commit the changes.

With the new fibrillate count, a cluster node is considered down if there is no heartbeat for 16 seconds. If network performance is poor, you may have to increase the fibirillate count to a higher value to avoid a DMS timeout.

Chapter 3. Cluster user IDs and user environments

In an HACMP cluster configuration, even a non-concurrent one, users often migrate from one system to another. If a failure has caused a node to leave the cluster, users may, without their knowledge, be signed on to a different system than the one they were signed on to the last time. It is an important customization requirement, then, that the same user environment and permissions be maintained across the various machines in the cluster. Users should be able to sign on to any machine in the cluster and have the same password, permissions, home directory, and other characteristics. This is not done by HACMP directly, so it must be implemented and managed separately. There is a need for some method of easily maintaining the same user environment across multiple machines in the cluster. In this chapter, we discuss several issues involved and some solutions to them.

3.1 User home directories

In a clustering environment, there are several options for deciding where to place your user home directories. These options include:

- Internal disks, as in normal AIX operations.
- External shared disks. By putting home directories on the shared external disks, the same home directories can be moved from system to system in failure situations.

3.1.1 Home directories on internal disks

If this is the choice made, user home directories are handled as they normally are with AIX. The /home file system, mounted on the hd1 logical volume of the rootvg volume group, houses the user home directories. The file system is physically located on one of the internal disks in the system.

If these user IDs are used throughout the cluster, this means that the same home directories must be created and maintained on all machines in the cluster that might, at some time, be serving the user. There is no common set of home directories for the whole cluster. This type of configuration is shown in Figure 40 on page 50.

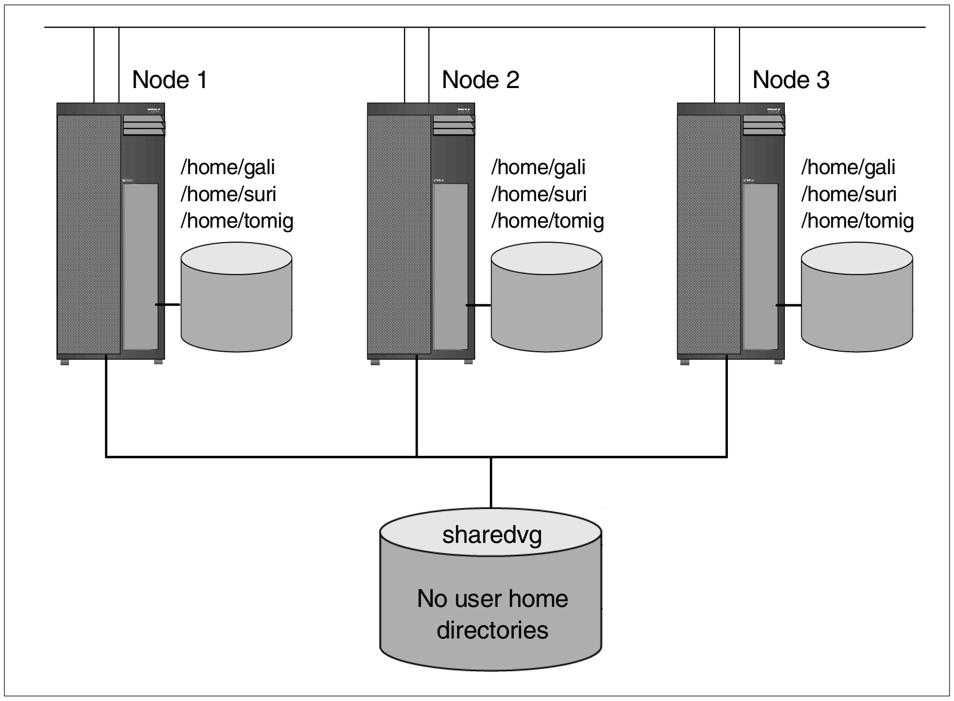


Figure 40. Home directories on node internal disks

With this alternative, you have multiple copies of the same home directories on different systems. Some client/server environments have the characteristic of highly-stable home directories or even a setup where all users or groups of users share the same home directory. Even in this case, you will want to have some method of keeping the home directory files automatically synchronized between systems. Chapter 4, “Keeping information synchronized in a cluster” on page 63 gives some examples of how to do this.

Maintaining home directories on the internal disks of each system in the cluster can be suitable if your user data is stable or if there is no changing data kept in these directories. This can also be suitable if you have a small number of user IDs that need to be usable throughout the cluster.

3.1.2 Common home directories on shared disks

Another option would be to use a single set of user home directories from all nodes. These directories would be located in a shared volume group on the shared disks, and they would be resources owned by one node. To implement this, you could remove the /home file system and /dev/hd1 logical volume from AIX on all nodes. Then, you would create a new shared logical volume in your shared volume group, and a new /home file system, mounted on that logical volume. Of course, you would import this information to any other nodes that could be taking over the file system in case of failure. In normal operations, all nodes other than the original server would mount this directory over the network using NFS. In case of node failure, one of the surviving nodes could break its NFS mount and take over the file system. Other nodes would continue to NFS mount the file system from the same address if IP address takeover were configured. This type of configuration is shown in Figure 41.

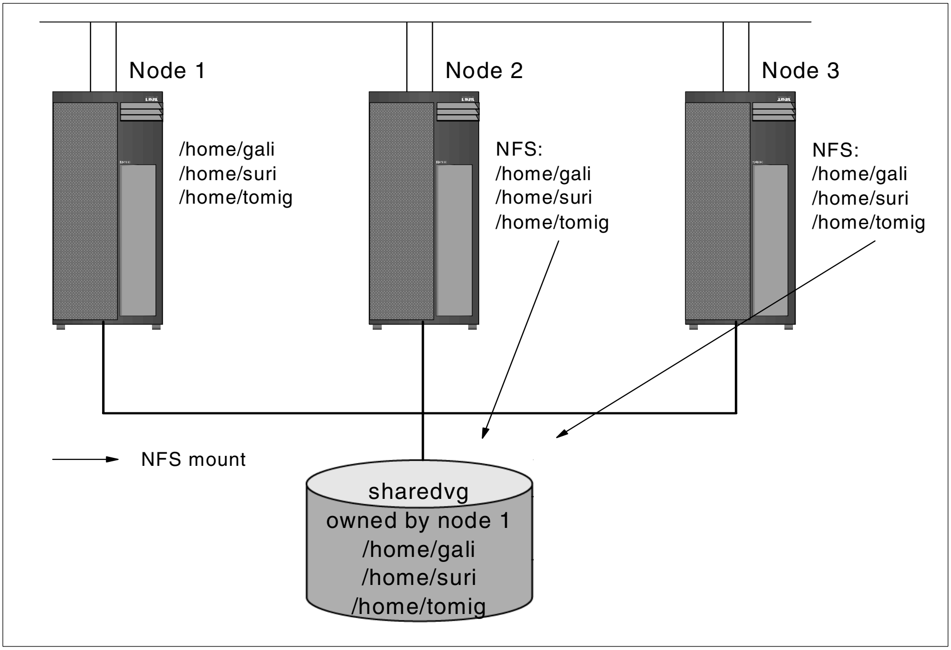


Figure 41. Common home directories on shared disks

This is a fairly simple approach that could be suitable in many situations. Since there is only one copy of the home directory contents, synchronization is not a concern. However, you must be aware of several possible issues that may or may not be a problem in your particular case:

- Using a home directory over NFS is, of course, slower than from a local disk.
- To implement this solution, you have to implement what we call cross mounting directories between nodes. This means that one or more nodes have NFS mounts to directories on another node.

- If you are using software that uses NFS locking on files in the home directories, you must be aware that HACMP does not currently have the function of preserving NFS file locks from one node to another in a node failure situation.

Given that these issues are dealt with, or do not apply to your environment, common shared home directories can be a good way to handle your user home directories.

3.1.3 Separate home directories on shared disks

There is a third possible solution to this requirement. In some clusters, multiple critical applications might each be served from different machines in the cluster. For instance, Node 1 has Application A defined as an Application Server resource that it owns, and Node 2 has Application B defined as an Application Server that it owns. In a two node cluster, these nodes might back each other up in what is called a mutual takeover configuration.

If each of these applications has its own separate and distinct set of users, who do not have to sign on and use the other application, it would be viable to have separate sets of home directories, each on a different shared volume group, and each owned by a different node in the cluster. In a node failure situation, the user home directories would be moved over to the backup machine along with the application data directories.

This concept could be expanded beyond two nodes to larger clusters as well. A picture of this kind of arrangement in a four node cluster is shown in Figure 42 on page 53.

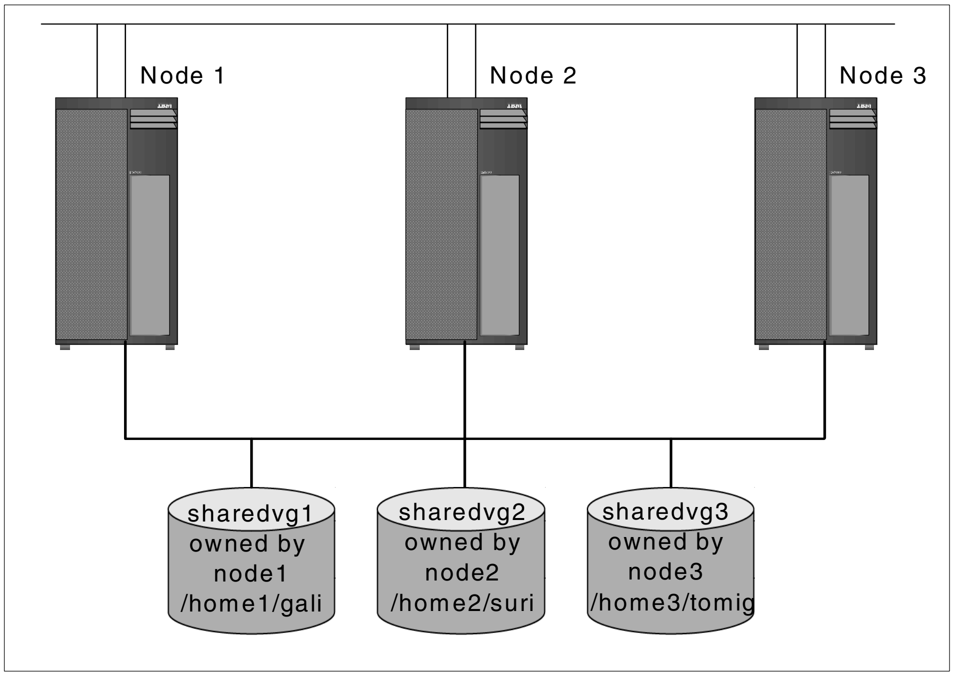

Node 1
Node 2
Node 3
sharedvg1 owned by node1 /home1/gali
sharedvg2 owned by node2 /home2/suri
sharedvg3 owned by node3 /home3/tomig

Figure 42. Separate home directories on shared disks

This approach has the benefits of only having one copy of each home directory on the shared disks (no need for synchronization between systems) and of not having to use NFS to access home directories. It can only be used effectively, however, in situations where each application has its own distinct set of users who do not need to access other applications.

3.2 Managing user IDs and passwords

Another important aspect of managing a cluster environment is to ensure that the same user IDS, with the same passwords, are available throughout the cluster.

There are several possible methods of keeping user IDs and passwords synchronized between nodes in the cluster. We will discuss three approaches here.

3.2.1 The rdist command

The `rdist` command provides a tool for periodically copying files from one machine to others. With this command, you can also keep system files synchronized between cluster nodes. User IDs and passwords are located in the /etc/passwd and /etc/security/passwd files. With the `rdist` command, you can propagate these files to all nodes every time you make changes to them. This can work whether you use SMIT or some other command to make the changes.

It is also possible to periodically check to see if certain files differ between different nodes. If they do, then rdist can be used to distribute them from what was decided to be the master copy. The checking and distribution function can be put in a shell script to be executed by cron. This method could be suitable in an environment where user IDs and passwords are not changed very often or where users from one system have infrequent need to sign on to the other system. It could also be suitable in a client/server environment, where users are actually signed on to client machines and connecting to a limited number of static user IDs on the cluster node servers.

One concern with using rdist or any other of the “r...” commands is security. Each of these commands require you to have /.rhosts files on each system. The use of these files can present a security exposure that is unacceptable for some installations.

There is a more in-depth discussion, and a usage example, of the `rdist` command in Chapter 4, “Keeping information synchronized in a cluster” on page 63.

3.2.2 Network Information Service (NIS)

While rdist provides a tool to distribute files (regular files, password files, or configuration files), NIS provides a more integrated and automated management system designed for managing common user IDs and passwords among multiple machines in a network. It can also be used to synchronize other information, such as the /etc/hosts file. NIS uses its own database of map files to keep its information. NIS is a client/server application, where synchronized information is kept on server machines, and it is requested by client machines. All systems that belong to an NIS domain have a consistent view of the information.

NIS is suitable for an environment where there are many active user IDs and systems. Because it provides an automated tool to keep user information synchronized in the cluster, it can be used to make the cluster look more like a single system.

Along with this benefit, NIS does have some drawbacks. NIS clients are strictly dependent on servers. If all the NIS servers are down, or a network failure causes a loss of connection to the client system, no users can sign on to the client system.

NIS also has a possible security problem. Unless Secure NFS is implemented (NIS is a facility of NFS), private map information, including passwords, are not encrypted going across the network. This opens up a security hole that could be exploited by potential intruders listening on the network.

An example of how to implement NIS in an HACMP cluster is given in the following section.

3.2.3 C-SPOC

C-SPOC is a facility that allows you to execute functions and commands on one or more cluster machines without having to log in explicitly and issue the commands there. This facility can be useful in a cluster configuration, because there are occasions when you will want to make the same changes on multiple machines. C-SPOC can increase the productivity of the systems administrator. You can make changes to all nodes as if it were a single system.

For initially setting up cluster user IDs and other systems administration tasks, C-SPOC can be quite useful. Once the cluster is in production, it could not be used by users to change their own passwords.

Similar to many other distributed tools, C-SPOC creates some security exposures that would not be there in a single system environment. This is a natural by-product of any facility that allows you to run administrative commands on remote systems. C-SPOC uses “r...” commands to perform the tasks on all nodes.

Issue the `smitty cl_users` command to start the SMIT C-SPOC interface. C-SPOC can perform the following user management tasks:

- Add a user to the cluster.
- Change/show cluster user characteristic.
- Remove a user from the cluster.
- List all users in the cluster.

Note

C-SPOC cannot change the user password.

C-SPOC is discussed in more detail in Chapter 3, “Cluster user IDs and user environments” on page 49.

3.3 Implementing NIS with HACMP

NIS provides a facility for maintaining common user IDs, passwords, and other configuration files across multiple machines in a network. NIS is used pervasively in the industry in non-high availability situations and can also be put to great advantage in a cluster to help manage a set of cluster user IDs. With its concept of master and slave servers, NIS has sort of an inherent redundancy, making it a useful tool in a cluster.

In an NIS domain, you can have one master server and one or more slave servers. Configuration information for all machines in the domain is stored in a set of maps on the master server. These maps are propagated to each of the slave servers, first when NIS is set up and then whenever a change is made to one of the maps. There are maps for user IDs and passwords, host names, and other information, if desired. Because the master server maintains the master maps for the domain, any changes to NIS maps should be made on the master server. The slave servers are capable of providing information to clients if the master server has failed but are not able to change NIS maps. Therefore, if the master server fails, you can continue having cluster users sign on and reference other information maintained by NIS, but you cannot make changes to it.

In a cluster, you can configure one node as an NIS master server and at least one other node as an NIS slave server. All nodes in the cluster can be either master or slave servers. Cluster clients can be NIS clients.

3.3.1 Preparing HACMP for NIS

HACMP needs to stop NIS services when it is changing adapter IP addresses. Therefore, you need to specify in your configuration whether you are running NIS or not. This example shows the necessary changes to the node environment. Remember, HACMP must be stopped and restarted on all nodes for changes to the node environment to become effective. Complete the following steps:

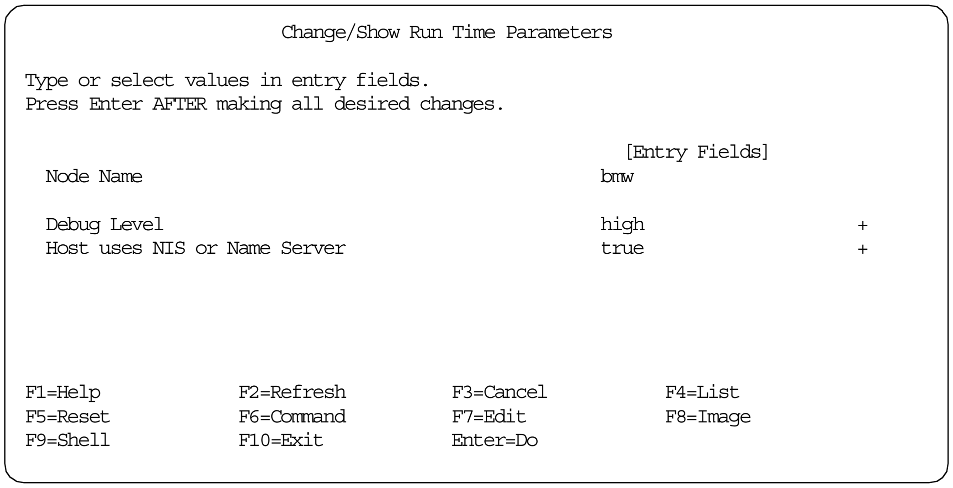
1. Stop HACMP on each node by entering the following command:

   ```
   smit clstop
   ```

   You will want to specify a graceful down.

2. Change the node environment run-time parameters for each node in **smit hacmp -> Cluster Configuration -> Cluster Resources -> Change/Show Run Time Parameters**.

   The panel shown in Figure 43 appears.



```
                         Change/Show Run Time Parameters

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
  Node Name                                           bmw

  Debug Level                                         high                 +
  Host uses NIS or Name Server                        true                 +





F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 43. SMIT panel for changing HACMP run-time parameters

   Set the Host uses NIS or Name Server field to **true** using the Tab key. Leave the other fields with their defaults, and press **Enter** to commit the change. Press **F10** to exit SMIT.

3.3.2 Configuring one node as an NIS master server

Before configuring NIS, you should check some prerequisites on all nodes:

- You must have TCP/IP running.
- You must have the portmap daemon running.
- NFS must be installed.
- The Cluster Manager must be running on the nodes you are configuring. This is because, if you are using IP address takeover, and you configure NIS before HACMP is active, NIS will be bound to the boot address, instead of the service address. This would make the server unreachable in normal operations.

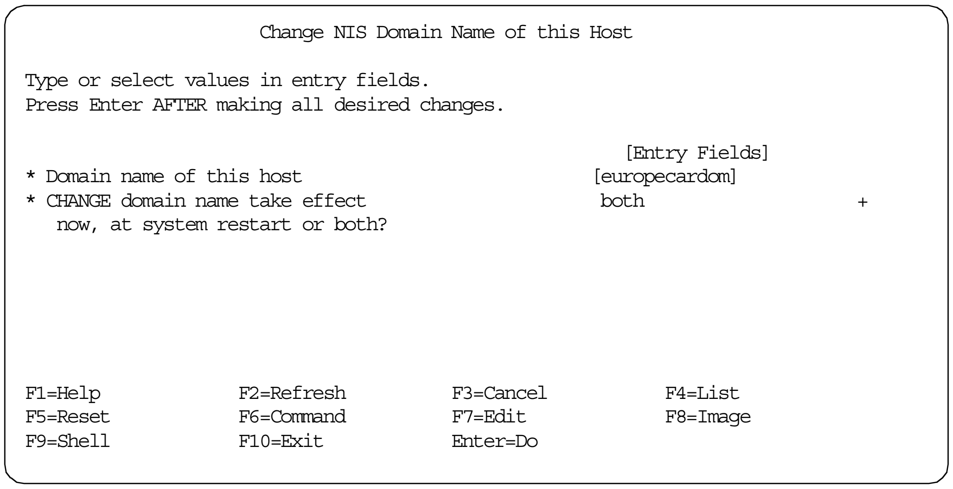
Log in as root to the node that will be your NIS master server, and perform the following steps:

1. Set the domain name by entering the following command:

```
smitty ypconfigure
```

2. Select **Change NIS Domain Name of this Host**.

   The panel shown in Figure 44 appears.

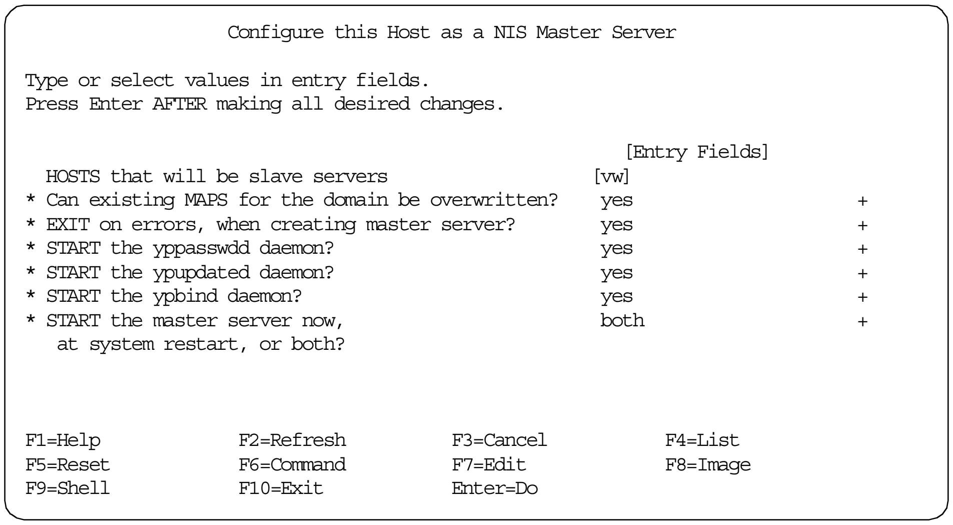


```
                      Change NIS Domain Name of this Host

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                     [Entry Fields]
* Domain name of this host                         [europecardom]
* CHANGE domain name take effect                    both                    +
   now, at system restart or both?







F1=Help              F2=Refresh           F3=Cancel            F4=List
F5=Reset             F6=Command           F7=Edit              F8=Image
F9=Shell             F10=Exit             Enter=Do
```

Figure 44. SMIT panel for changing NIS domain name

   Fill in the domain name you want to use, for example, europecardom. Leave the other fields with their defaults. Press **Enter** to commit the changes. Then press **F3** to return to the Configure/Modify NIS panel.

3. Configure your node as an NIS master server by selecting **Configure this Host as a NIS Master Server** from the SMIT panel. You can see the appropriate SMIT panel on Figure 45 on page 59.



```
                    Configure this Host as a NIS Master Server

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                         [Entry Fields]
  HOSTS that will be slave servers                    [vw]
* Can existing MAPS for the domain be overwritten?     yes                    +
* EXIT on errors, when creating master server?         yes                    +
* START the yppasswdd daemon?                          yes                    +
* START the ypupdated daemon?                          yes                    +
* START the ypbind daemon?                             yes                    +
* START the master server now,                         both                   +
   at system restart, or both?



F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 45. SMIT panel for configuring the NIM master server

Fill in the host name of those nodes that will be NIS slave servers to the appropriate field, for example, vw. Set the following fields to `yes`:

- START the yppasswdd daemon?
- START the ypupdated daemon?
- START the ypbind daemon?

Leave the other fields with their default settings.

4. Do a final check of the /etc/inittab file. Make sure the NFS subsystem is still configured to start at the run level a, and not 2. Change it back to a if it is something else.

3.3.3 Configuring other node(s) as NIS slave servers

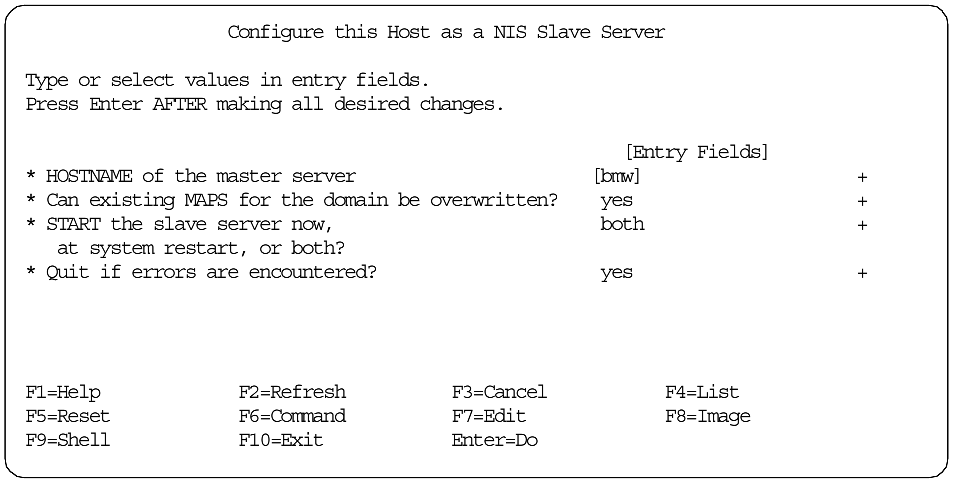
On each node that will be an NIS slave server, do the following:

1. Check to see if the /var/yp/binding directory exists. If it does, remove the directory and its contents. This removes any previous NIS server binding information and allows the slave to bind to the master server.
2. In the SMIT menus, change the NIS domain name as we did when configuring the NIS master server.
3. Configure your node as an NIS slave server by entering the command:

```
smitty ypconfigure
```

4. Select **Configure this Host as a NIS Slave Server**.

The panel shown in Figure 46 will appear.



```
                    Configure this Host as a NIS Slave Server

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
* HOSTNAME of the master server                      [bmw]                   +
* Can existing MAPS for the domain be overwritten?    yes                    +
* START the slave server now,                         both                   +
   at system restart, or both?
* Quit if errors are encountered?                     yes                    +




F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 46. NIS slave server configuration in SMIT

Fill in the host name of the NIS master server that you just configured, for example, bmw. Leave the other fields with their defaults. Press **Enter**. This should configure your slave server and start the ypbind and ypserv daemons. Press **F10** to exit SMIT.

Repeat these steps for each node that will work as an NIS slave server.

3.3.4 Configuring client systems for NIS

Here, we show you how to configure client systems to use NIS services. Complete the following steps:

1. In the SMIT menus, change the NIS domain name as we did when configuring the NIS master server.
2. Configure your system as an NIS client by entering the command:

   ```
   smitty ypconfigure
   ```

3. Select **Configure this Host as a NIS Client**.

   You can take the default value to start now and at system restart. Press **Enter** to commit the changes and then **F10** to exit

Repeat these steps for every AIX system that will be an NIS client.

3.3.5 Managing NIS maps

All changes to NIS maps should be done on the master server. User passwords are the one exception to this rule. They can also be changed from other servers or clients by using the `yppasswd` command, as described in Section 3.3.6, “Changing a user password” on page 62.

After the changes have been made on the master server, NIS maps are propagated to the slave servers in one of three ways:

- Immediately by using the `yppush` command from the master server or by changing to the /var/yp directory and issuing the make command.
- Automatically every few minutes if you are running the ypupdated daemon on your master server.
- Automatically at specified intervals by using the cron daemon from each of the slave servers.

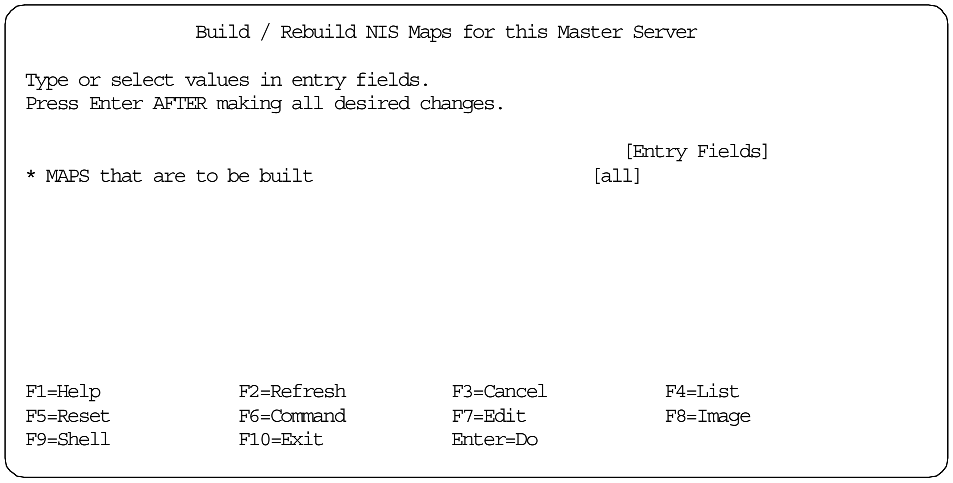
For example, if you want to add a new user ID to your cluster:

1. First, you create the user ID and password with SMIT in the normal way.
2. Then use one of the three methods above to propagate the maps to the slave servers.

   There is a way to carry out the first method by using SMIT, as in the following example:

   ```
   smitty mkmaps
   ```

   You can see the SMIT panel in Figure 47.



```
                Build / Rebuild NIS Maps for this Master Server

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                           [Entry Fields]
* MAPS that are to be built                             [all]









F1=Help              F2=Refresh           F3=Cancel            F4=List
F5=Reset             F6=Command           F7=Edit              F8=Image
F9=Shell             F10=Exit             Enter=Do
```

Figure 47. SMIT panel for building NIS maps for the master server

3. Select **all** for all NIS maps and press **Enter** to commit your changes. This will rebuild the maps for the master server and propagate them to all slave servers. Thus, all clients will get this new information.

3.3.6 Changing a user password

You should generally make all changes to NIS information on the master server. The one exception to this rule is changing passwords. You can change your password from any NIS client or server machine. If you use the standard `passwd` command in any machine other than the master server, it will fail. This is because, if you are using NIS, your user ID is no longer managed from the /etc/passwd file.

NIS provides a function to change your password, even from client systems, by issuing the `yppasswd` command.

This command behaves exactly like the normal `passwd` command, except that it makes its changes to the NIS map in the master server. Thus, the change is effective for all machines in your NIS domain.

The `yppasswd` command can only be used when the master server is up.

Chapter 4. Keeping information synchronized in a cluster

The cluster nodes must share some information to ensure the high availability of the systems and of the running applications. The data in the shared volume group are common to the nodes and HACMP itself has the Global Object Data Manager (GODM) facility, which keeps the HACMP-related ODM entries synchronized between cluster nodes. Nevertheless, you might need to synchronize other system components. We focus on two main synchronization items:

- How to keep synchronized files in non-shared file systems.
- How to keep synchronized the time clocks.

4.1 File synchronization

The file synchronization can be fulfilled by the AIX command `rdist` whenever the files on the cluster nodes are exactly equal. Sometimes, you need only a partial file synchronization, that is you want a part of the file on the cluster nodes to be the same and another part to be different. We supply the /etc/security/passwd synchronization as an example of the latter need.

4.1.1 The rdist command as a synchronization tool

The purpose of the rdist command is to distribute identical copies of files to multiple hosts. It preserves the owner, group, mode, and modification time of files that it distributes. The rdist command can receive its instructions for which files to distribute and to which hosts from several sources, including:

- A default distribution file, named distfile, in the home directory of the user issuing the command
- A different distribution file, specified in the command arguments with the -f flag
- Command line arguments that supersede the information in the distribution file
- Command line arguments that act in themselves as a small distribution file

The distribution file specifies the files to copy, destination hosts for distribution, and operations to perform when updating files. The name of the distribution file is specified on the command line when we issue the rdist command. This way, we can use different timings for different files by using separate distribution files for each set. For instance, we want to distribute user files, which tend to be more volatile, more often than system files, which tend to be more static. The rdist command requires that a .rhosts file be configured on each host. We can distinguish between user file distribution and system file distribution. In the former case, we are assuming that we have decided to put user home directories on the internal disks, rather than on shared disks. Because users might choose to sign on to either node, we have to be able to distribute the user files in either direction when they change. When we are updating files from Node A to Node B and from Node B to Node A, we have to make sure we are not trying to update in both directions at the same time. Therefore, we must make sure that the distribution timing in each systems crontab file is different. Also, we must have some method of controlling system clocks in each node. Clocks can be synchronized with the timed subsystem. We discuss more about the timed subsystem below. The -y parameter of the rdist command prevents a more recent copy of a file from being overwritten by an older version. For the system files, we will dedicate one node as the master server. This means that we will make all changes to the system files only in one node, to be distributed from it to the other nodes. This way, we do not have to worry about relative timing of updates for system files.

In the following, we make up an example of system file distribution and we provide a sample script that makes it possible to keep files synchronized even if we only change the file inode attributes. This function is not supported by the standard rdist command. Let’s choose to distribute the files /etc/hosts, /.rhosts, /etc/passwd, and /etc/security/passwd. The /etc/security/passwd file contains passwords in encrypted form. The encryption does not become a problem, since the encrypted passwords are actually in ASCII format. Since the encryption algorithm is identical on all AIX machines, the password will be resolved the same way on all nodes. First, we create the distribution file distfile.system on the cluster node ford that is the distribution master server:

```
HOSTS=(dodge)
FILES=(/etc/hosts /.rhost /etc/passwd /etc/security/passwd)
${FILES} -> ${HOSTS}
```

The other node, dodge, is in the HOST list, and the FILES list contains the system files we want to distribute. The rdist command scans the distribution file and interprets the file or directory names as well as host names of other nodes. Then it compares file sizes and modification times between the corresponding files in the different nodes. If the version in the other node has a different time stamp or is a different size, it copies the file to that node. This rule is subject to the use of the -y parameter, which makes sure that only older file versions get overwritten. The standard rdist command does not care about the update time stamp of the file. Therefore, if you change file attributes, for example, with the `chmod` command, the `rdist` command does not recognize the change. Because of this shortcoming of the rdist command, we have included a sample script in Figure 48 on page 66 that checks if the update time stamp is newer than the modification time stamp. If this is the case, the script touches the file to update the modification time stamp. After this change, the rdist command will update this file to the other nodes. The rdist_update script is shown in Figure 48 on page 66. You can also get it from the additional material to this book provided at:

```
www.redbooks.ibm.com
```

Note

The rdist_update script changes the file modification time stamp. This could conflict with other application using the file time stamp.

The sample script uses standard distribution files. The script does make some assumptions, however, that the standard rdist command does not make:

- The filenames must be assigned to the variable FILES in the distribution file.
- Full pathnames must always be used for files or directories. Do not use wildcards when defining filenames.
- The host names are handled as with the standard rdist command.

If you have a need to avoid these restrictions, you need to make your own changes to the sample script. This can be done by changing the method of collecting file names to the DISTFILES variable.

```
#!/bin/ksh
#
# Name: rdist_updated
#
# Update files with rdist to other nodes also if the i-node is updated since
# the last modification.
#
# Author: Teppo Seesto, IBM Finland
#
############################################################################
# collect files to check from distfile
# Note! Do not use wildcards when defining files in distfile
# Use always FILES variable to define used files in distfile
#
DISTFILES=`awk ' / FILES/,/\)/' $1 | awk '{ORS=" " ; print }' | \
awk '{F S = " [()]"; print $2}'`
FILELIST=$DISTFILES

for dfile in $DISTFILES
do
  # check if file is directory
  # if it is extend FILELIST with contents of the directory
  if [ -d $dfile ]
then
     FILELIST=$FILELIST" "find $dfile -name "*" -print
  fi
done

for file in $FILELIST
do

  # check i-node update time and modify time
  UPD=istat "$file" | grep updated | awk '{print $4" "$5" "$6" "$7}'
  MOD=istat "$file" | grep modified | awk '{print $4" "$5" "$6" "$7}'
  # compare times
  # if updated after modification touch the file
  if [ "$MOD" != "$UPD" ]
  then
    touch $file
  fi
done
# do rdist normally after touching files
rdist -f $1 -y
```

Figure 48. rdist_update script

4.1.2 Updating the password as a synchronization issue

User password management is a good example of a partial file synchronization. We have two cluster nodes and we want users to be able to change their password. The password change must be propagated from a node to the other where the same user is defined. Let’s assume we have a two node cluster and the cluster node names are ford and dodge. The users idris, ole, vicktor, and andrea are commonly defined users. The malevolent user andrea changes his password on node ford, which is the present production node. After the next takeover, the user andrea cannot log in to the system and run his application. Then, we supply a script that keeps the passwords synchronized for the user, but does not affect the system user passwords, for example, root bin, and so forth.

Note

The script shown in Figure 50 on page 69, Figure 51 on page 70, and Figure 52 on page 72 overwrites the system file /etc/security/passwd. You must be sure that it is compatible with your system environment before it runs.

The script's purpose is to cross check the files /etc/security/passwd of the two cluster nodes, for example, ford and dodge, and to synchronize the two files with the latest, updated user password. The script should run on a regular basis on both nodes but the running time in each system must be different. First, the syncpwd.sh script identifies the common users and then it checks the password and the updating password time of each user. If the updating password time on the remote system is different from the updating password time on the local system, the local password is overwritten with the most recent one. Then, the most recent password for each user is propagated to each cluster node and the /etc/security/passwd files are properly synchronized.

```
#!/usr/sbin/ksh
#
################################################################################
# The script syncpwd.sh is used to update user password on a system comparing  #
# its file /etc/security/passwd with the same file on another system. The      #
# script works only if the .rhosts file of the remote system contains the IP   #
# label of the local one.                                                      #
#                                                                              #
# You have to customize some parameters below before the script runs.          #
#                                                                              #
# The present script overwrite the file /etc/security/passwd. Then you must be #
# sure it does not conflict with other application before you issue it         #
#                                                                              #
# This script is not supported and it is meant only as an example.            #
#                                                                              #
#                                                                              #
# Date: 9/2/2000                                                               #
#                                                                              #
################################################################################
#
# Select a working directory for temporary files and backup files
#
savedir=/tmp
#
# Set the node IP label where the script is running
#
localnode=dodge
#
# Set the remote node IP label
#
remotenode=ford
#
# Choose the user-id below which the user passwords are not affected
#
userid=200
#
#
# Now the parameters and variables are set. Here the main starts
#
#
# Initialize a variable and clean a file
#
chgpwd=0
test -s  $savedir/upduser.list  && rm $savedir/upduser.list
#
# Create a copy of the file /etc/security/passwd
#
date=`date +%j%H%M`
cp -p /etc/security/passwd  ${savedir}/passwd.$date
#
# Purge the old copy of the file  /etc/security/passwd
#
find $savedir -name 'passwd.*'  -atime +30 -exec rm {} \;
```

Figure 49. Shell script syncpwd.sh: Part 1 of 3

```
#
# Start a cycle for each user defined on both nodes
#
for user in `cat $savedir/$remotenode.list`
do
if `grep $user $savedir/$localnode.list >/dev/null 2>&1`
then
#
# Get the encrypted password of the same user on $remotenode and $localnode
#
grep -p $user $savedir/passwd.$localnode | awk -F"=" '/password/ {print $2}'| re
ad pwdlocalnode
grep -p $user $savedir/passwd.$remotenode | awk -F"=" '/password/ {print $2}'| r
ead pwdremotenode
#
# Get the last update of the password on $remotenode and $localnode
#
grep -p $user $savedir/passwd.$localnode | awk -F"=" '/lastupdate/ {print $2}'|
read lastlocalnode
grep -p $user $savedir/passwd.$remotenode | awk -F"=" '/lastupdate/ {print $2}'|
 read lastremotenode
#
# Compare the password and the last update; if the $remotenode password has
# been more recently update we propagate it also on $localnode together with
# the lastupdate time
#
   if [[ $pwdlocalnode != "$pwdremotenode" &&  $lastremotenode -ge $lastlocalnod
e ]]
   then
#
# Count how many user passwor are updated
#
        (( chgpwd += 1 ))
#
# Record the user whose password is updated
#
echo $user >> $savedir/upduser.list
```

Figure 50. Shell script syncpwd.sh: Part 2 of 3

```
#
# Update the file /etc/security/password
#
cat /etc/security/passwd | awk -vuser="${user}:" -vpwd=$pwdremotenode -vlast=$la
stremotenode  ' BEGIN{pwdmarker=0; lastmarker=0}
                !/password/ && !/lastupdate/ {
                                               print $0
                                             }
                $1 == user { pwdmarker=1
                             lastmarker=1
                           }
                /password/ {if (pwdmarker == 0) {
                                                 print $0
                                               }
                            if (pwdmarker == 1) {
                                  printf "%+19s%s\n","password = ",pwd
                                  pwdmarker=0
                                               }
                           }
                /lastupdate/ {if (lastmarker == 0) {
                                                     print $0
                                                   }
                              if (lastmarker == 1) {
printf "%+21s%s\n","lastupdate = ",last
                              lastmarker=0
                                                   }
                              }'  > /newpasswd.tmp
        cp -p /newpasswd.tmp /etc/security/passwd
   fi
fi
done
#
# A smal report is produced in the directory $savedir
#
echo "Report on the running of the script syncpwd.sh on node `hostname`">$savedi
r/REPORT
echo "`date`">>$savedir/REPORT
echo>>$savedir/REPORT
echo "The follwoing $chgpwd user password has been updated:">>$savedir/REPORT
echo>>$savedir/REPORT
cat $savedir/upduser.list >>$savedir/REPORT 2>/dev/null
#
# Save old report and purge the old reports
#
cp -p $savedir/REPORT $savedir/REPORT.$date
find $savedir -name 'REPORT.*'  -atime +30 -exec rm {} \;
```

Figure 51. Shell script syncpwd.sh: Part 3 of 3

4.2 Synchronizing the clocks in a cluster

In a distributed application environment, it is important for systems to show the same time of day on their clocks. This requirement is even more critical in an HACMP cluster environment. For instance, in a failover situation, if the clock on the takeover node is behind that of the owner node, it could create problems on tasks, such as database recovery, using logs on the shared disks. The takeover machine's database software would be trying to recover transactions that were technically in the future according to its time clock.

Also, when you are monitoring or troubleshooting your cluster, it will be easier for you if all the clocks in your cluster are synchronized. This way, the log files from different machines can be interpreted chronologically.

4.2.1 The timed daemon

The timed daemon synchronizes one machine's clock with those of other machines on the local area network that are also running the daemon. The timed daemon slows the clocks of some machines and speeds up the clocks on other machines to create an average network time.

When the timed daemon is not started as a master time server, it locates the nearest master time server and asks for the network time. Then, the system uses the `date` command to set the clock to the network time. The system then accepts synchronization messages periodically sent by the master time server and makes corrections to the system's clock.

When the timed daemon is started as a master time server, the machine polls each of its LANs to determine which networks have master time servers. The machine becomes a master time server on the networks that do not already have a master time server. The machine becomes a submaster time server on the networks that already have a master time server. The timed daemon creates the /var/adm/timed.masterlog file, which contains a log of the deltas between the local clock and its clients' clocks.

4.2.1.1 Implementing time serving in your cluster

In an HACMP/6000 cluster, you should have at least two time servers to gain highly available time services. You can start the timed daemon as a server in all nodes so that you have time services whenever you have at least one server up.

The timed daemon, therefore, provides its own high availability function. Therefore, HACMP/6000 does not have to explicitly take care of that daemon. If the master time server dies, a new master time server is elected from the submaster time servers on the network. This does take a few minutes to happen, since it is done on the next messaging cycle. The messaging cycle between machines is done every four minutes. The `timedc` command enables you to select which submaster time server becomes the new master time server.

The timed daemon can be controlled by using the System Resource Controller (SRC), SMIT, or the command line. The daemon is not started by default.

To start the timed daemon as a time server from the command line, issue the following command:

```
root@mickey:/> startsrc -s timed -a "-M"
```

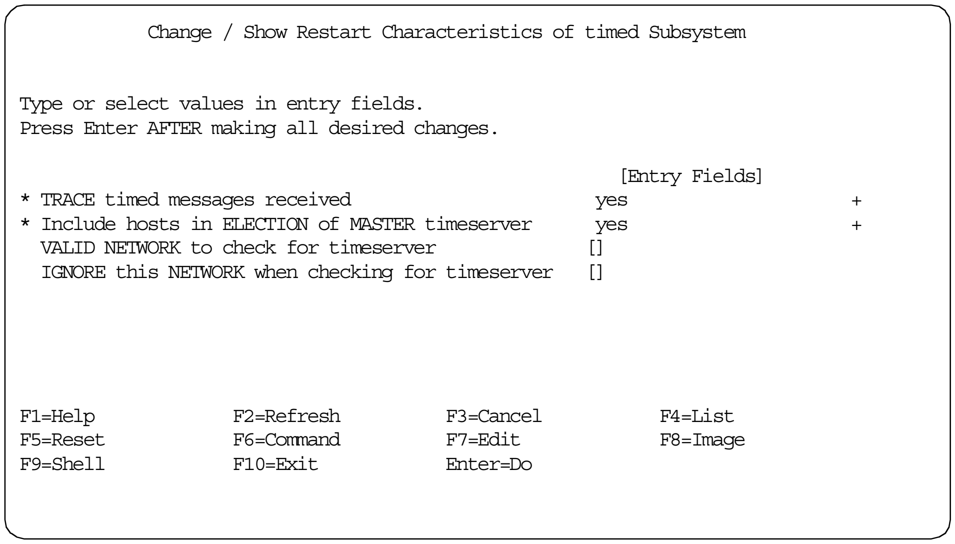
To start the timed daemon as a time server and to have it start automatically on future system reboots, issue the command:

```
root@mickey:/> chrctcp -S -c timed -f master='yes' -f trace='yes'
```

This command will change the /etc/rc.tcpip file. In this format, the command will also start tracing the timed daemon. Leave out the `-f trace='yes'` option if you don't want to trace. You can do this also through SMIT by issuing the command:

```
root@mickey:/> smit timed
    Start Using the timed Subsystem
```

Select **BOTH** to start the timed daemon, both now and at system startup. Press **F3** twice to return to the Timed Subsystem panel. Select **Change / Show Restart Characteristics of timed Subsystem**, and you will get the panel shown in Figure 52.



```
             Change / Show Restart Characteristics of timed Subsystem


Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
* TRACE timed messages received                       yes                    +
* Include hosts in ELECTION of MASTER timeserver      yes                    +
  VALID NETWORK to check for timeserver              []
  IGNORE this NETWORK when checking for timeserver   []






F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 52. SMIT panel for the timed daemon

Select with the Tab key whether you want to trace your timed daemon or not. In the Include hosts in ELECTION of MASTER timeserver field, select **yes** to make this machine a time server. In the other fields, you can select the networks whose time serving you potentially want to control. The default is to include all networks to which the system is connected. Press **Enter** to commit your changes and then **F10** to exit SMIT.

To start the timed daemon as a client system from the command line, issue the following command:

```
root@mickey:/> startsrc -s timed
```

To start the timed daemon as a client system with each reboot, issue the command:

```
root@mickey:/> chrctcp -S -c timed
```

This command will change the /etc/rc.tcpip file. Alternatively, you can do this through SMIT by issuing the command:

```
root@mickey:/> smit sttimed
```

Select **BOTH** to start the timed daemon, both now and at system restart. Press **Enter** to commit your changes and then **F10** to exit SMIT.

Chapter 5. NFS in a cluster

The cluster nodes can be an NFS server. In order for NFS to work as expected on an HACMP cluster, you must choose an NFS maintenance strategy that matches the application needs. The HACMP scripts offer one solution for managing NFS, but sometimes, you may want a different one. In this section, we review the standard HACMP way of maintaining NFS and we present an alternative approach to bypass the problems that could occur using the standard HACMP method.

5.1 Implementing NFS using the HACMP facilities

HACMP allows you to define NFS exported file systems as resources that are protected and migrated within the cluster. A file system that is exported by one node can be acquired and exported by another node. The NFS clients experience nothing except a delay during the failure detection and takeover time. HACMP also allows what we call NFS cross mounts: one cluster node mounting a file system, through NFS, from another cluster node. If this is done, a cluster node that was originally an NFS client can become an NFS server after the failure of its partner node.

However it is implemented, running NFS in an HACMP cluster introduces some special considerations. This section provides information about the proper assignment of major device numbers, exporting NFS file systems, and issues involved in cross mounting.

5.1.1 Major device numbers

As part of the mounting process for an NFS file system, the server system passes something called a *file handle* to the client system. The file handle is used by the client system thereafter to reference the file, and it is made up of four pieces of information:

1. The major device number of the special file (logical volume) containing the file system in which the file resides.
2. The minor number of the special file.
3. The i-node number of the file within the file system.
4. A generation number that is used to check whether the file handle is out of date. The generation number for the file is incremented whenever the file's i-node is updated.

If you are NFS mounting file systems from shared disks, all this information should be equal in each node that can take over these disks. After takeover, the file handle held by an NFS client should still be valid on the new server. In a clustered environment, the minor number, i-node number, and generation number would all be the same on each node because they are self-contained within the logical volume. The only one of the four items that could be variable is the major device number. The logical volume inherits its major device number from the major device number of the volume group that contains it. It is essential, then, that any shared volume group that will contain NFS exported file systems have the same major device number on all nodes in the cluster.

Note

While the requirement to make any shared volume group have the same major device number throughout the cluster is only critical when NFS is to be used, it is a very good practice to follow generally. It is sometimes hard to predict, in the beginning, all the ways that shared data will be used. Having a common major device number prepares you for the use.

When a volume group is first created, or when it is imported, you have the option of specifying a major device number to be assigned to it. If a major device number is not specified, the system will assign the lowest unused number. In a cluster of systems, this next lowest unused major device number will not necessarily be the same, and in fact, almost never is. Major device numbers are assigned to categories of devices, such as TTYs, printers, disks, and so on. It is easy to see how differences in machine configurations can cause differences in the major device numbers that are used.

In a cluster, the task then is to find a major device number that is commonly available on all machines in the cluster and then to specify it to be assigned to a new shared volume group. You can find this by using the `lvlstmajor` command. For instance, we might issue this command on node ford:

```
ford::root:/>lvlstmajor
46,47,49...
```

The output from this command indicates that the major device numbers 46, 47, 49, and higher numbers are unused and available on this machine. Issuing the same command on node dodge may give us the following response:

```
dodge::root:/>lvlstmajor
31,...
```

Here, we find out that 31 and any number higher is unused and available on this machine.

Repeat this on every node that might take over the shared volume group. Find a common number that is unused in every node. This is a valid major device number to use for your new shared volume group.

You can change the major device number of a volume group as follows:

1. Find out what major device numbers are free on each node, using the method we used before. For example, you find 46 is free.
2. Deactivate the volume group, if it is active, with the command:

   ```
   ford::root:/>varyoffvg fordvg
   ```
3. Export the volume group by entering the command:

   ```
   ford::root:/>exportvg fordvg
   ```
4. Then import the volume group again with a new major device number, for example, 46, by entering the command:

   ```
   ford::root:/>importvg -y fordvg -V 46 hdisk4
   fordvg
   ```

   The major device number for fordvg is now 46.

The major device number for any volume group can be changed by simply exporting it and then reimporting it with the desired major device number. This can be done even if the volume group has just been created and is only known on one system. Remember that the `exportvg` command does not touch the volume group definitions in the Volume Group Descriptor Areas (VGDAs) on the disks themselves. Therefore, after the export, this information is still intact and can be used to reimport the volume group.

5.1.2 NFS file system exports with HACMP

Standard NFS servers use the /etc/exports file to list the directories available for mounting by clients, the clients allowed to access them, and any access restrictions. The shell scripts provided with HACMP do not use /etc/exports for file system exports that are defined as resources. When HACMP exports file systems for NFS mounting, it calls the /usr/sbin/cluster/utils/cl_export_fs script. The cl_export_fs script calls the `exportfs` command with the `-i` flag, passing the file systems you have specified for export as parameters. HACMP stores these export file system names in the ODM file /etc/objrepos/HACMPresource. The file systems are referenced by the EXPORT_FILESYSTEM variable in the HACMP shell scripts.

The -i flag of the exportfs command means that it will ignore contents of the /etc/exports file. The APAR IY05357 introduces a new feature to manage the NFS server in a cluster. The /usr/sbin/cluster/exports file use capability was added but is optional. This was accomplished by changing the cl_export_fs script to use this file, if it exists, and by changing the clconfig script to synchronize this file to all cluster nodes. In this step, you add the directories of the shared file systems to the exports file. In particular, you can use this capability to export shared file systems of the cluster to an NFS client which is outside the cluster itself. Complete the following steps for each file system you want to add to the exports file:

1. Enter the `smit mknfsexp` fastpath to display the Add a Directory to Exports List screen.
2. In the EXPORT directory now, system restart or both field, enter `restart`.
3. In the PATHNAME of alternate Exports file field, enter:

   `/usr/sbin/cluster/etc/exports`
4. Add values for the other fields as appropriate for your site, and press **Enter**. HACMP for AIX uses this information to update the /usr/sbin/cluster/etc/exports file.
5. Press **F3** to return to the Add a Directory to Exports List screen, or **F10** to exit SMIT.

Repeat steps 1 through 4 for each file system you want to export. You can also edit the /usr/sbin/cluster/etc/exports file and add entries as in the usual file /etc/exports. After that you can synchronize the cluster and the file is propagated to every cluster nodes.

Eventually, you can specify NFS exporting options, or client hosts to give root access, by modifying the cl_export_fs script at the points where the `exportfs` command is issued. There are two instances of the `exportfs` command in the current version of the script. The first instance is to export file systems where you have hosts requesting root access. The second one is for exports without root access. If you wanted, you could use the /etc/exports file as in a typical NFS environment by removing the `-i` flag from the `exportfs` commands in the cl_export_fs script. This may not be such a good idea because the /etc/exports file is static. Its contents do not change in response to cluster events. Therefore, not only would you have to maintain multiple /etc/exports files around the cluster, but the contents of the file might not always match what should be exported in a particular cluster state.

In any case, if you modify the cl_export_fs script in any way, you must take care to maintain these changes. The cl_export_fs script is an HACMP utility, which can be overwritten by HACMP PTFs. You have to be sure to make a copy of this and any other modified scripts before applying PTFs, take careful note of which scripts were modified by the PTF, and reintegrate your changes, if necessary.

5.1.3 Cross mounted NFS file systems

An NFS cross mount is a situation where one cluster node is an NFS client of another node. This is often done to allow users across multiple nodes to have access to the same data, without running concurrent access. If you do implement NFS cross mounts, there are several issues to be aware of. These have special importance in a node failure when a surviving node might have to take over and mount the file system it previously had mounted through NFS.

Here is a typical problem situation:

1. Node ford owns and exports a file system resource that is NFS mounted by node dodge. Node ford has that file system defined as a takeover resource which should be acquired and exported if node ford fails.
2. Node ford fails. Node dodge starts to acquire the disks and file systems it has defined as takeover resources, including the file system it had NFS mounted.
3. Node dodge cannot mount this file system locally until it has unmounted it with NFS.
4. If there are any processes with open files in the NFS mounted file system, or any user whose current working directory is in it, the unmount will not complete.

HACMP/6000 includes a utility named cl_nfskill, called by the script cl_deactivate_nfs, which is run at takeover time on node dodge. This utility detects all processes with open file descriptors on the file system and kills them. After it has killed the processes, it does a forced unmount of the file system. Since the processes that had files open in the NFS mounted file system have been killed, the forced unmount will work, even with the server down. You will not have to wait for any timeout.

If there are users on node dodge, with processes using file systems that were NFS mounted from the failed node ford, these processes will be killed during takeover. For instance, if a user is signed on to node dodge, with their home directory mounted from the failed node (a cluster user with home directory on shared disk), and their home directory is their current directory, the user will be logged out when node dodge takes over for node ford. The user’s shell process will have been killed to allow the NFS unmount to proceed. Without this, node dodge would not be able to take over that file system.

This user will have to reconnect again after the takeover has completed.

5.1.4 A new method for the NFS cross mounted file systems

In HACMP for AIX 4.3.1, a new method for NFS cross mounting provides a solution for problems with unmounting the NFS mounted file systems on takeover attempts or reintegrations. This design uses an NFS mount point different from the local mount point and requires that the file system be NFS mounted on the node which also has the file system locally mounted. All applications must reference the file system strictly through the NFS mount. With the NFS mount point different from the local mount point, you do not need to unmount the NFS mount on takeover or reintegration. With this method, the same changes would be made to the usual cross mounting procedure. The following example clarifies the new configuration for the NFS cross mounting in a cascading resource group.

The node ford and node dodge are the two participants to a cascading resource group named fordrg. The applications that work on node ford and node dodge need data contained in the path /sharedfs_nfs. The file system /sharedfs is defined on both nodes and it contains the application data. On both nodes, we create the NFS mount point /sharedfs_nfs and we update the resource group forvg filling in the Filesystem to NFS Mount field as follows:

```
/sharedfs_nfs;/sharedfs
```

This is <nfs mount point>;<local mount point>.

In the case of both cluster nodes running, the scenario is:

- /sharedfs locally mounted on node ford
- /sharedfs NFS mounted over /sharedfs_nfs on node ford
- /sharedfs NFS mounted over /sharedfs_nfs on node dodge

If node ford fails, node dodge would just mount the file system locally and export it. The scenario is:

- /sharedfs locally mounted on node dodge
- /sharedfs NFS mounted over /sharedfs_nfs on node dodge

It is highly recommended that you use this new method to avoid possible unmounting problems with the NFS mounts. More detail can be found in /usr/lpp/cluster/doc/release_notes.

5.2 Implementing NFS bypassing the HACMP facilities

Some applications may experience problems in an HACMP environment where NFS also is implemented. Let's take a two node cluster. The node ford and node dodge have a shared file system, /sharedfs, that belongs to a cascading resource group fordrg. The node ford is the primary node and it also exports the file systems /sharedfs on node dodge. As ford fails, dodge obtains the file system /sharedfs to be locally mounted. Nevertheless, a nasty application prevents /sharedfs to be unmounted as an NFS file system. HACMP uses the `mount` command to check if the file systems obtained in a takeover are already mounted and it finds an entry for the file system /sharedfs; therefore, it does not mount the file system again.

You can avoid this scenario by completing the following steps:

1. Review Section 5.1.3, "Cross mounted NFS file systems" on page 79 on how to define a volume group that contains an NFS-exported file system.
2. On both nodes, change the file system mount point, for example:

   ```
   ford::root:/>umount /sharedfs
   ford::root:/>chfs -m /sharedfs_mnt /sharedfs
   ```

3. Then, on node ford, remove the old mount point /sharedfs and create a symbolic link /sharedfs that points to /sharedfs_mnt:

   ```
   ford::root:/>rmdir /sharedfs
   ford::root:/>ln -s /sharedfs_mnt /sharedfs
   ```

4. On node dodge, remove the mount point /sharedfs and create a soft link as well:

   ```
   dodge::root:/>rmdir /sharedfs
   dodge::root:/>ln -s /sharedfs_mnt /sharedfs
   ```

5. Then, NFS export the file system /sharedfs_mnt on both nodes.
6. If the cluster is stopped, you do not have to mount the file system /sharedfs_mnt and it is mounted on node ford at the cluster restart provided that you put it in the cascading resource group fordrg. On the other hand, you have to NFS mount it on node dodge after it is made available by the NFS server.

This configuration uses soft links to make the data available to the application. The applications look for the data through the path /sharedfs while the system manages the mounting and unmounting operations through the mount point /sharedfs_mnt.

As the node ford fails, the node doge takes over. You can issue the `mount` command on node dodge after a takeover and you will obtain the output displayed in Figure 53.

```
dodge::root:/>mount
  node       mounted        mounted over    vfs       date        options
-------- ---------------  ---------------  ------ ------------ ---------------
         /dev/hd4         /                jfs    Feb 07 11:18 rw,log=/dev/hd8
         /dev/hd2         /usr             jfs    Feb 07 11:18 rw,log=/dev/hd8
         /dev/hd9var      /var             jfs    Feb 07 11:18 rw,log=/dev/hd8
         /dev/hd3         /tmp             jfs    Feb 07 11:19 rw,log=/dev/hd8
         /dev/hd1         /home            jfs    Feb 07 11:20 rw,log=/dev/hd8
ford     /sharedfs_mnt    /sharedfs        nfs3   Feb 11 12:59 soft,intr
         /dev/sharedlv    /sharedfs_mnt    jfs    Feb 11 13:02 rw,log=/dev/svg1log
         /dev/tsmlv       /tsmshr          jfs    Feb 11 13:02 rw,log=/dev/svg1log
```

Figure 53. The output of the mount command

The node dodge performs as its own NFS server and assumes this function also for other NFS clients. The file system /sharedfs_mnt is locally mounted, and on top of it, the file system /sharedfs is NFS mounted. If you don’t like this (we do not), then you can customize cluster events to manage the NFS mount and unmount in order to access the /sharedfs_mnt data through the soft link rather than through an NFS mount point. The post-event for the node_up_local_complete event takes care to mount the NFS file system on node dodge as shown in Figure 54 on page 83.

```
#!/usr/bin/ksh
#
# Post node_up_local_complete
#

set -x


if [[ `/usr/es/sbin/cluster/utilities/get_local_nodename` = dodge ]]
then
      /usr/es/sbin/cluster/utilities/clgetactivenodes  -n ford
      rc=$?
  if [[ $rc -eq 2 ]]
     then
      mount ford:/sharedfs_mnt  /sharedfs &
  fi
else
      exportfs -a

      /usr/es/sbin/cluster/utilities/clgetactivenodes  -n dodge
      rc=$?
  if [[ $rc -eq 2 ]]
     then
      rsh dodge mount ford:/sharedfs_mnt  /sharedfs &
  fi
fi

exit 0
```

Figure 54. The shell script for the node_up_local_complete post-event

The pre-event to the node_down_remote event shown in Figure 55 prevents the file system /sharedfs to be NFS mounted after a takeover on the node dodge.

```
#!/usr/bin/ksh
#
# Pre node_down_remote event
#

set -x

if [[ `/usr/es/sbin/cluster/utilities/get_local_nodename` = dodge ]]
then
umount -f /sharedfs &
fi

exit 0
```

Figure 55. The shell script for the node_down_remote pre-event

Using these event customizations, the mount command returns the output shown in Figure 56 when you issue it on node dodge after a takeover.

```
dodge::root:/>mount
  node       mounted        mounted over    vfs       date        options
-------- ---------------  ---------------  ------ ------------ ---------------
         /dev/hd4         /                jfs    Feb 07 11:18 rw,log=/dev/hd8
         /dev/hd2         /usr             jfs    Feb 07 11:18 rw,log=/dev/hd8
         /dev/hd9var      /var             jfs    Feb 07 11:18 rw,log=/dev/hd8
         /dev/hd3         /tmp             jfs    Feb 07 11:19 rw,log=/dev/hd8
         /dev/hd1         /home            jfs    Feb 07 11:20 rw,log=/dev/hd8
         /dev/sharedlv    /sharedfs_mnt    jfs    Feb 11 13:02 rw,log=/dev/svg1l
og
         /dev/tsmlv       /tsmshr          jfs    Feb 11 13:02 rw,log=/dev/svg1l
og
```

Figure 56. mount command output on node dodge after a takeover

The file system /sharedfs_mnt is locally mounted and the applications get the data via the soft link /sharedfs that points to /sharedfs_mnt. In Chapter 8, “HACMP and SAP R/3” on page 109, you can find an example where this method is implemented.

5.3 Hint: NFS server and multiple network environment

Almost every cluster has more than one network, but you may need to use only one specific network for your NFS server. You can force the cluster nodes to use the network you choose for NFS mounting. You must define a resource group that manages the sole IPAT of the NFS server and you must associate the file systems to be exported to such a resource group. In this way, the NFS mounting is bounded to the network associated to the resource group.

Chapter 6. Highly available print queues

Printing is a natural by-product of running a commercial UNIX system. Applications generate reports that are printed and reviewed. In some cases, an application will take a significant amount of time to scan the data and do the calculations, sort, and other tasks to produce the report that will be printed. In these cases, the loss of a print job that has been generated because of a system failure can be a major inconvenience, if not a direct negative impact to the business. For this reason, there is some interest in the high availability community for solutions to protect the print jobs that have been generated from loss in a node failure. This chapter outlines procedures and gives examples for implementing print queue protection for the AIX print queuing system.

6.1 General AIX printing concepts

There are several general AIX concepts dealing with the printing subsystem and other subjects that should be reviewed at the beginning.

Note

For the purposes of this discussion, we use the terms *queuing* and *spooling* interchangeably.

6.1.1 Components of the AIX 4.3 print queuing subsystem

It is important to understand the way in which AIX 4.3 implements print queuing before trying to implement our highly available printing solution. The following reviews some terms connected with the printing subsystem in AIX 4.3.

Printer	This is the actual hardware device on which output is printed. Technically, in AIX V4.3, a printer is the combination of the real printer device and the printer device driver software.
Print Queue	A print queue is a named list to which print jobs are sent to await the availability of a printer device. A print queue can be served by one or more printers. The settings of the print queues are stored in the /etc/qconfig file plus the associated files in the /var/spool/lpd/pio directory.

Qdaemon	The qdaemon is a process that runs in the background. It is usually started automatically with the `startsrc` command when the system is powered on from a line in the /etc/inittab file. The qdaemon keeps track of the print requests in the /var/spool/lpd/qdir directory and ensures that the jobs are sent to the proper printer at the proper time. It also keeps track of the status of the printers and stores printer usage data for system accounting purposes. The status and accounting information is held in the /var/spool/lpd/stat directory.
Printer Backend	The printer backend is a collection of programs invoked by the qdaemon subsystem to manage a print job that is queued for printing. The default printer backend is called *piobe* (printer input/output backend).
Printing Commands	The main print spooling command for AIX is `enq`. You can invoke this command directly to queue a print job, or use one of three front-end commands: `lp`, `lpr`, and `qprt`. A print request issued by one of these commands is first passed to the `enq` command, which places the information about the file in the queue for the qdaemon to process. This queue is located in the /var/spool/lpd/qdir directory. If the job is not a file (that is, pipe output of a command to enq), a real file is created in the /var/spool/qdaemon directory, which contains the data to be printed. The information in the /var/spool/lpd/qdir file then points to the file in /var/spool/qdaemon.

6.1.2 Printing from a file copy

By default, most of the printing commands for AIX print directly from the file in its file system location, rather than making a temporary copy of the file first and then printing from that copy. The exception to this rule is the `lpr` command, which, by default, makes a copy of the file to be printed in the /var/spool/qdaemon directory and then prints from that copy.

As you will see later, our high availability printing solution involves putting the spool directories on the shared disk so they can be moved to a backup system if necessary. Since application files to be printed are typically from file system locations on the internal disks of the nodes, making a copy of the file in the spool directory, and printing from that copy, will be the preferred approach. This way, the print file is still available even if the node where it was printing fails.

As previously mentioned, if you are using the `lpr` command to do your printing, the default command will make the file copy. However, if you are using the commands `enq`, `lp`, or `qprt` to do your printing, you will want to use the `-c` option of the command to make this copy in the /var/spool/qdaemon directory.

6.1.3 How to attach your printer in a cluster

Printers that are directly attached to a cluster node become unavailable and unusable if that node fails. This includes printers that are connected to the systems in the traditional ways, such as by parallel or serial interfaces. In recent years, methods of attaching printers directly to a network instead of to a single machine have been introduced. This not only makes it easier to share a single printer among multiple machines in a network, but it is also very advantageous in a highly available cluster. The printer remains available and accessible, even if specific machines in the cluster fail.

Here are some of the ways in which you can connect a printer so that it is not directly attached to the cluster:

- Use a printer with an integrated network adapter to attach it directly to your network.

  The IBM 4039 Laser Printer is one example of a printer with direct network attachment capability.

- Attach your printer to a RISC System/6000 on the LAN.

  This alternative involves attaching your printer to a remote system, which is running the lpd subsystem to make it a remote print server. Each of the cluster nodes defines remote print queues pointing to queues on this print server machine. Using this approach, the printer is available to all nodes in the cluster, but the failure of the print server machine will cause the unavailability of the printer.

Note

The IBM 4033 LAN Connection is another device that is used to attach printers to a network in many installations. In our testing, we found that this connection was not suitable for a high availability configuration, since for this configuration, we need to be able to define identical direct queue definitions to the printer from multiple cluster nodes. The 4033 only allows one machine to have a direct queue definition to the printers attached to it. When a second machine defines a direct queue definition, the first machine that had the existing definition loses its ability to print to the device.

It could still be possible to use such a connection by defining a third machine as its print server, as in the first option in the list above. In this solution, each of the cluster nodes would have remote print queues defined to queues on that print server.

In the following example, we implement standard TCP/IP remote print queues, as described in the first option above. Remote printing is implemented according to client/server architecture. The remote print jobs pass through the print spooling subsystems of both the client (cluster node) and the server systems.

When the print request is submitted, it is queued in the same way as a local print job. The qdaemon process on the client machine processes the print request as it would process any job queued locally. The difference, in the case of a remote print job, is that the queue device that is used to process the job uses the rembak backend program rather than piobe.

The rembak program transmits the print job to a remote print server. The lpd daemon on the print server receives the print job sent to it and places it on the appropriate local queue. The print job is now processed just like any other local job.

6.1.3.1 Spooling directory placement

One of the major elements in planning a highly available print queuing system is the placement of spooling directories on the shared disks.

Usually, each cluster node has its own copy of AIX stored on an internal disk attached to an internal I/O adapter. The application software programs also reside on the internal disk. The shared external disks store mission-critical data.

Now let's consider our print queues. If the print spool directory is on the internal disk, access to that spool directory will be lost if the node fails. If we put the spool directory on the shared disks instead, and a node detaches from the cluster, the backup node can acquire the shared disks. The backup node can provide exactly the same view of disk services, and the queued print jobs can still be saved. As part of our solution, we change the standard configuration of AIX such that two spooling directories, /var/spool/lpd/qdir and /var/spool/qdaemon, are located on our shared disks instead of the internal disks.

We create these two file systems as you do other shared file systems in HACMP—first creating two logical volumes and then creating the two file systems on top of them on one machine. Then, the shared volume group is imported to the other machines to ensure that the ODMs on all cluster nodes contain information about these shared file systems.

In case of failure, one of the surviving nodes takes over the shared disks, activates them, verifies the state, mounts these two file systems, then restarts the qdaemon and the print queues.

Note

To protect against a disk failure and data loss, we suggest you use AIX Logical Volume Manager (LVM) to mirror all shared logical volumes. If a disk fails, the machine simply continues operations using the other mirrored disk.

6.2 An HACMP implementation of the AIX spooling subsystem

The following section describes an implementation of highly available print queues using the AIX 4.3 spooling subsystem. As we said in the previous discussion, using the AIX spooler, we can do print queue protection only in one direction. Figure 57 on page 90 shows the cluster for this example.

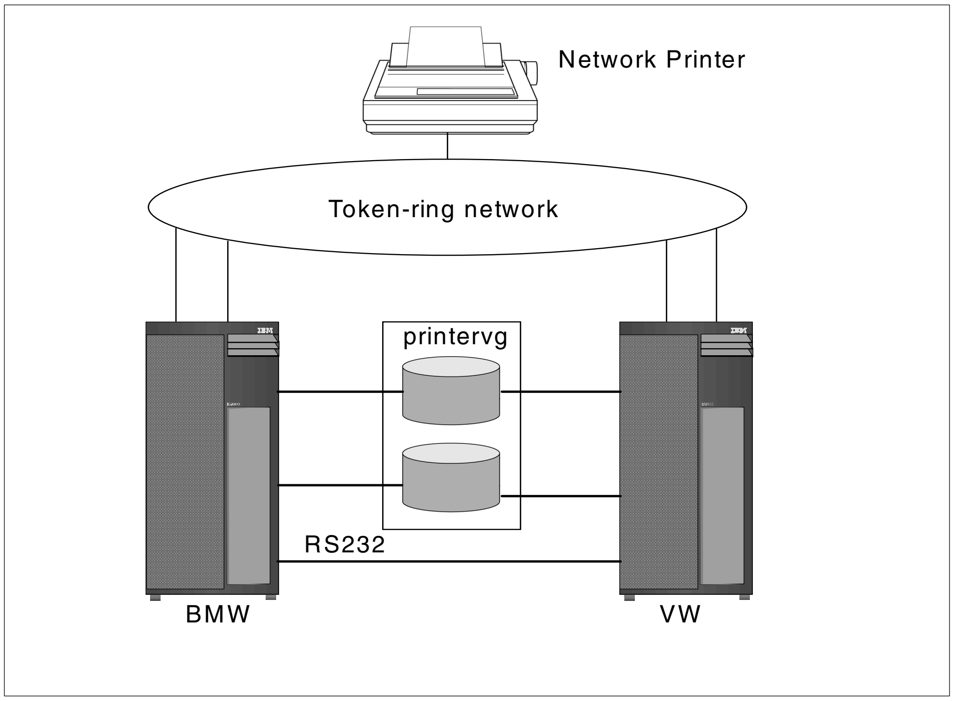


Figure 57. Cluster for highly available print queues

An HACMP cluster is set up in Hot Standby mode between two nodes, named bmw and vw. Node bmw is the server and node vw is the backup. The cluster is configured for IP address takeover with interface labels as shown in Table 1.

Table 1. Cluster network interfaces

Interface	**Node bmw**	**Node vw**
Boot	bmw_boot	vw_boot
Service	bmw	vw
Standby	bmw_sb	vw_sb

Additionally, the cluster is set up with two shared disk strings between the two nodes. We have a shared volume group, called printervg, defined on the shared disks. The network printer device called audi is attached to the network. The printer queue that we will be printing to from our cluster nodes is called pass.

6.2.1 AIX setup activities

The AIX setup activities will include defining remote print queues on the cluster nodes and the proper placement of the spooling directories on the shared disks.

6.2.1.1 Configuring a remote print queue on each cluster node

In our example, the print server (network printer) will be the machine called audi, and the print clients will be the cluster nodes bmw and vw.

Note

Your network printer server's remote queue setup procedure may differ from this example. Please refer your printer manual for the setup instructions.

We set up remote queues for the server node, bmw, and the backup node, vw, with the following steps:

1. Create the remote queue:

   ```
   smit mkrque
   ```

2. Select your printer type from the list.
3. Select the appropriate type of remote printing.

   You can see the appropriate SMIT screen in Figure 58 on page 92.

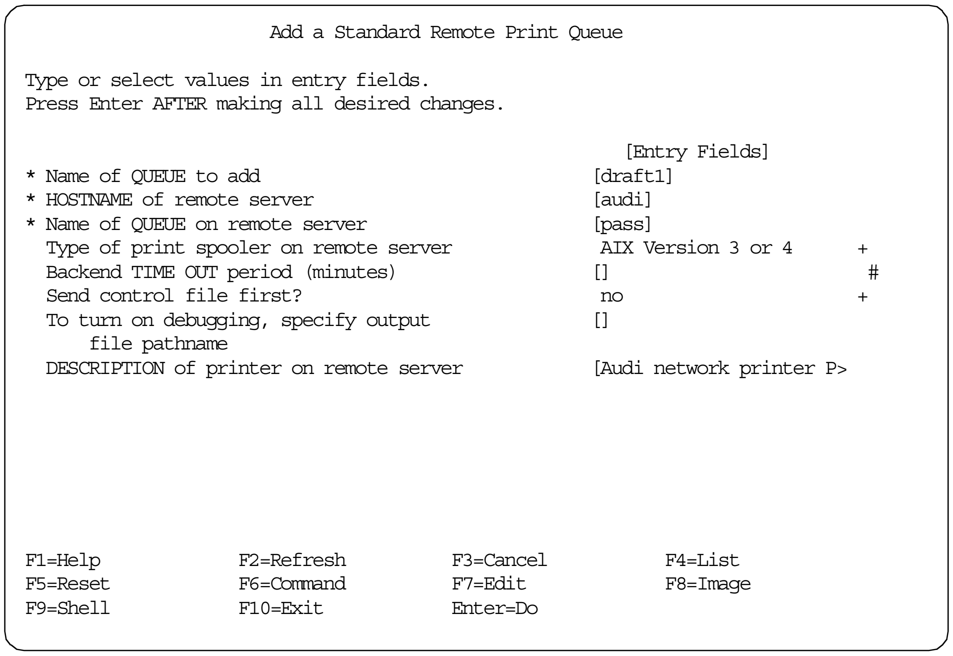

```
                        Add a Standard Remote Print Queue

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
* Name of QUEUE to add                                [draft1]
* HOSTNAME of remote server                           [audi]
* Name of QUEUE on remote server                      [pass]
  Type of print spooler on remote server               AIX Version 3 or 4      +
  Backend TIME OUT period (minutes)                   []                        #
  Send control file first?                             no                      +
  To turn on debugging, specify output                []
      file pathname
  DESCRIPTION of printer on remote server             [Audi network printer P>




F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 58. Printer queue definition panel in SMIT

On this input panel, you must enter the queue name, the destination host name, and the name of the queue on the printer server. In the example above:

- The queue named **draft1** has been selected.
- The destination host **audi** has been selected.
- The remote queue **pass** has been selected. This is the name of the queue on the print server machine that will accept the remote print jobs.

4. Repeat the pervious steps on the backup node.

Important

Make sure that the two node’s remote print queue definitions are consistent. Otherwise, the print queue of the first node cannot be taken over by the second node.

Now we have finished our print queue setup at both nodes. The printer on audi is ready to accept jobs from the nodes bmw and vw.

6.2.1.2 Printing on remote printers

AIX supports a variety of print commands and print systems of different heritages. You may use any of the printing commands, or SMIT, to print to a remote printer.

You can use `lpr`, `lp`, `qprt`, or `enq` to issue a print job from the command line. One difference between these commands is that the command `lpr`, by default, prints from a saved temporary copy of the file in the `/var/spool/qdaemon` directory. The other commands do not do this by default, but can do so by using the -c option.

Important

In order to provide highly available printing service, it is necessary, when we issue the `lp`, `qprt`, and `enq` commands, that we add the `-c` option to the command to make a copy of the print file in the directory /var/spool/qdaemon. This directory will be on the shared disks, so it will be movable from one node to another.

6.2.1.3 Configuring shared file systems

We will need to add spooling file systems that can be used by both nodes. Create the following two shared file systems:

- /var/spool/qdaemon
- /var/spool/lpd/qdir

Please ensure that the underlying volume group is not activated automatically at system restart. Also, give unique names for the related logical volumes and jfslog. Set the mount automatically at system restart option to `false` for both file systems. Finally, import the shared volume group to the backup node with the proper major device number.

6.2.2 Customizing HACMP

We are assuming that you already configured and synchronized your cluster topology with the appropriate network settings.

6.2.2.1 Create an application server for qdaemon

Here, we are adding an application server to stop and restart the qdaemon after the file systems containing the print files and queue records have been moved over. This is a necessary step in the process since the qdaemon must be restarted after the print files have moved from the other system in order for it to see them.

Always ensure that a qdaemon subsystem is stopped before starting it, since a second qdaemon will adversely affect the printing subsystem. If a second qdaemon is accidentally started, it may be stopped with the `kill -9` command. This method, however, should not be used to stop the qdaemon under normal conditions.

The application server definition is as follows:

Server Name:	printer
Start Script:	/usr/es/sbin/cluster/events/start_printer
Stop Script:	/usr/es/sbin/cluster/events/stop_printer

The application server startup script (/usr/es/sbin/cluster/events/start_printer) is as follows:

```
#!/bin/ksh
# Application server startup script
# This script restarts the spooler subsystem
stopsrc -g spooler
sleep 5
startsrc -g spooler
# Start the draft1 queue
/usr/bin/qdmin -U draft1
```

The application server stop script (/usr/es/sbin/cluster/events/stop_printer):

```
#!/bin/ksh
# Application server stop script
# This script stops the spooler subsystem
stopsrc -g spooler
```

6.2.2.2 Configure resource groups

The next step is to configure the necessary resources. We need one cascading resource group to implement the highly available printer queues. Create the resource group with the following settings:

Resource Group Name:	printqrg
Node Relationship:	cascading
Participating Node Names:	bmw, vw
Service IP Label:	bmw

Filesystems:	/var/spool/qdaemon /var/spool/lpd/qdir
Filesystems Consistency Check:	fsck
Filesystems Recovery Method:	sequential
Filesystems to Export:	
Filesystems to NFS mount:	
Volume Groups:	
Concurrent Volume groups:	
Raw Disk PVIDs:	
AIX Connections Services:	
AIX Fast Connect Services:	
Application Servers:	printer
Highly Available Communication Links:	
Miscellaneous Data:	
Inactive Takeover Activated:	true
9333 Disk Fencing Activated	false
SSA Disk Fencing Activated:	false
Filesystems mounted before IP configured:	false

Finally, verify and synchronize the cluster resources. We have now finished our highly available printer queue setup. After starting the HACMP on both nodes, you can submit printer jobs to the network print server. If the primary node fails, the backup node can takeover the first node’s unfinished print out files and finish the printing.

6.3 Summary

The preceding is an example of implementing protection of print queues in a Hot Standby type of HACMP configuration. Because of the fixed location of the spool directories, this is the only way we were able to accomplish highly available printing with the AIX spooling subsystem.

Chapter 7. Planning for application integration with HACMP

Applications are the most important part of a cluster. HACMP provides the facility to automatically launch an application at the end of a cluster startup cycle and to stop the application at the beginning of cluster shutdown.

Integrating applications with an HACMP cluster could vary from simple to complex depending what you want to achieve. One thing to remember, whatever you want to achieve must be supported by the application itself, since HACMP only provides an infrastructure to make an application highly available.

This chapter helps you integrate an application in the HACMP cluster environment. It includes an explanation of the characteristics of applications suitable for an HACMP environment and how to prepare the application to work in an HACMP cluster.

7.1 Applications suitable for an HACMP environment

Each application has its management features and behaves differently from one another. An application best suits the HACMP cluster environment if it has the following features:

- Automatic recovery
- Unattended startup and shutdown
- Host independent

7.1.1 Application with automatic recovery

The application should be able to handle automatic recovery after a system failure to ensure the application data integrity. The feature is required, because in the event of a system failure, the application is stopped abruptly, and records and tables are not properly closed. For example, an RDBMS application should be able to restart after a crash and perform a data recovery, such as a roll backward.

An application that requires manual intervention after a system crash is not suitable in an HACMP cluster. The application cannot work in a failover situation. When the application is restarted on the backup server, the startup will fail because it requires manual intervention to clean up the database before it can be restarted again.

Fortunately, most RDBMS software, such as IBM UDB, Oracle, Informix, Sybase, and Ingress, support automatic recovery. Therefore, when buying business-critical applications, make sure that considerations are given to this feature.

7.1.1.1 Clearing application locks

Some applications create lock files when the applications are started. The files are written into a directory that is defined during installation. It is a simple method of tracking an improper shutdown of an application. It is also a means of ensuring the integrity of the application data.

When the application is shutdown normally, the lock files are cleared as part of the shutdown process. However, if the application is killed by a system administrator or if the system crashes, the lock files are left in the lock directory.

At the next application startup, the startup process detects the existence of lock files, therefore, requesting the system administrator to clean up the locks before proceeding with the application startup.

In an HACMP cluster environment, such application behavior results in a failure in the application server startup process, preventing the application from being started automatically during a failover.

However, in applications that requires the removal of lock files, you can integrate the application into an HACMP cluster by removing the lock files before starting the application. The removal can be included in the startup script of the application server.

Note

Refer to the application administration manual for the exact procedures to clean up the application lock files because it may impact the data integrity of the application.

7.1.1.2 Using manual recovery

If you still want to use the application server in an HACMP cluster, do not include the application server in a resource group since it would jeopardize the overall availability of the HACMP cluster. You should define only the file systems in the resource group, but the application should be started manually. Upon a completion of the system failover process, perform a manual recovery, and start the application on the backup node manually.

The manual startup might cause you more down time, but the time is much less than in a stand-alone environment. If the error notification is configured to notify the system administrator of the system failure, the manual recovery process can be performed promptly.

7.1.2 Application with unattended startup and shutdown

HACMP launches an application server as part of the HACMP startup or shutdown sequence. The application server executes a predefined script to start or to stop an application. At this point, the application is started or stopped in the background without any intervention.

The non-interactive startup or shutdown is important to ensure that the application is launched or stopped with the failover sequence to minimize the startup or failover time, thus, minimizing down time.

If an application requires a system administrator to use a menu as a startup procedure, the application does not fit very well in a high availability environment. The interactive methods prevent the application to be integrated with an HACMP application server because an application server fires up an application in the background.

However, in some applications where the interactive menu requires simple responses, you might be able to bypass by using a command redirection.

For example, Figure 59 illustrates a startup screen of a sample application that requires a manual intervention.

```
# start_mysample
                Starting Up the MYSAMPLE databse


        Please select the mode you wish to start the databse:

                1.  Online Startup
                2.  DB Maintenance
                3.  Abort the startup

                Option: 1

        Please wait.  The MYSAMPLE DB is starting........
        MYSAMPLE DB is Online
#
```

Figure 59. A sample screen of an interactive application startup

By selecting from the startup menu, the application is started. You can use several methods to pass the value 1 to the startup program. One of the methods is to use an echo command to pass the required value to the program. The following is how you can use echo:

```
# echo 1 | start_mysample
```

In a multiple response situation, you can use a redirection such as the following example:

```
# start_mydb < mysamp_response
```

The content of mysamp_response is shown in Figure 60.

```
# cat mysamp_response
mount sampdb
start sampdb
exit
```

Figure 60. Content of a sample input file

If the application requires interactive startup and shutdown, and you still want to have HACMP running on the application server, we suggest that you exclude the interactive application from all HACMP resource groups and control the application by manual startup and shutdown as suggested in Section 7.1.1, “Application with automatic recovery” on page 97. As previously mentioned, the reason is that if HACMP is waiting for input to the application, the resource group will hang and HACMP will never start.

7.1.3 Application without host dependency

Some applications are designed to be dependent on the host mainly as a security feature. There are several types of host dependency that require further customizing to allow the applications to work in an HACMP cluster environment.

7.1.3.1 Run-time host dependency

When an application is started, it checks for the host name of the system that it is running. If the application detects a different host name, it produces error messages and refuses to start. It could be designed to ensure data integrity. However, in an HACMP environment, an application should be able to run on any node with a different host name to allow smooth failover of the application.

The host name definition is done during the software installation. Some applications can be configured to be host name dependent or host name independent. Others can only be configured to run on the same host.

The host name of a backup server should be changed to the host name of a production server to accommodate such applications. The host name change can be performed within the start script of the application server or it can be done by customizing a cluster event.

7.1.3.2 Software license host dependency

Some application vendors provide license keys for applications based on the system identification of a system. The license key is only valid on the system that is specified during the license key application. In an HACMP environment, the application cannot start after a failover because the license key is no longer valid on the backup server.

If you are using such applications, it is recommended that you consult the software vendor to issue a secondary license key for the backup server. You may have to explain the environment to avoid paying an additional license fee. Unfortunately, no customizing can be done to get around the license restriction. The second license is required to get the application to work in an HACMP cluster.

7.2 Configuring a shared application in a cluster

Normally, an application is installed and configured on a production node. Once the application is successfully configured on a production node, the next step is to configure the application on a backup node.

7.2.1 Installing an application on a production server

The following are recommendations for the configuration of applications on a backup server:

1. Prepare a plan for an application installation. Refer to the software installation manual for the proper installation guidelines.
2. Create an external shared volume group.
3. Create shared file systems.
4. If you have a choice, install the application on the shared file systems. Most third-party software allows the systems administrator to specify a path to install the software. IBM software, which uses the installp installation method, automatically installs the software in the /usr/lpp directory in rootvg.
5. Configure your application. If possible, put the configuration files in the shared file system.
6. Start your application and create your database.

7. Test the application.

These are guidelines on how to setup the applications. However, you must refer to the application installation guide for the specific requirements of the applications.

7.2.2 Configuring the application on a backup server

Once you have completed the installation on the production server, you have to make the application available to the backup server. Therefore, you have to make the shared disks available to the backup server.

The following are steps you should perform on the production server:

1. Stop the applications.
2. Unmount all shared file systems.
3. Varyoff all shared volume groups.

Once the shared resources are released on the production server, you can start making the resources available on the backup server:

1. Varyon all shared volume groups on the backup server.
2. Mount all shared file systems on the backup server.
3. If the application was installed in the internal disks on the production server, you have to install the application again. If you installed the application on the shared disk, you may skip this step.
4. Configure your application. In most cases, you need to make sure the configuration files are in proper directories. However, there are some applications that require the whole setup procedure to be performed step by step as in the production server. In any case, always consult the installation guide to ensure that you have done the right procedure.
5. Start your application. If your application can start successfully, you have completed the setup of your application.

7.2.3 Testing an application failover

Once you have configured the applications on both the production and the backup servers, you have to create a startup script and a stop script for the application. The scripts must be defined in an application server that is a part of a resource group.

Before you test the cluster failover, you should a perform a manual failover to test all the scripts that you have written work according to your needs. The following are the steps that you should take to do a manual failover test of your application.

On the production system:

1. Configure the service address on the primary adapter.
2. Varyon all shared volume groups.
3. Mount all shared file systems.
4. Execute your application start script.
5. Test the application.
6. Execute your application stop script.
7. Unmount all shared file systems.
8. Varyoff all the shared volume groups.
9. Configure the production boot address on the primary adapter.

If the test is successful on the production server, you should now move to the backup server and proceed with the following steps:

1. Configure the production service address on the standby adapter.
2. Varyon all shared volume groups.
3. Mount all shared file systems.
4. Execute your application start script.
5. Test the application.
6. Execute your application stop script.
7. Unmount all shared file systems.
8. Varyoff all the shared volume groups.
9. Reset the standby address on the standby adapter.

You have completed the manual failover of your cluster. By doing a manual failover test, you have more control over the failover steps. You should fix any problems before proceeding to the next steps. When you are doing the HACMP failover, you have to wait till the end of the startup of the failover sequence before you can make any modifications, and the process repeats until you have corrected any errors.

Therefore, the manual failover test helps you to trouble shoot any application problems before performing the automatic HACMP failover test.

Note

The forced shutdown of an HACMP cluster is not available in HACMP/ES 4.3.1. Therefore, to recover from a shutdown failure, you have to reboot the node.

7.3 Sample application installation

For the purpose of exhibiting the process of configuring an application in a shared HACMP environment, we go through an installation process of the Oracle software.

Note

This installation example does not go through a step-by-step process for installing the Oracle software. Refer to the product installation manual for the software installation procedure. We only provide guidelines on how to make the application available on two different servers.

The following are the steps taken in configuring Oracle in a shared environment:

1. Create an oracle (UID=1000) and dba (GID=1000) group. The oracle user home directory is /oracle, which is a shared file system.
2. Log in as oracle.
3. Install Oracle in the /oracle file system. The installation process creates several configuration files and updates the /etc/inittab file.
4. Start Oracle.
5. Start the Oracle Listener.
6. Configure the database in the shared disks.
7. Stop Oracle.
8. Unmount all shared file systems.
9. Varyoff all shared volume group.

On the backup server:

1. Create an oracle (UID=1000) and dba (GID=1000) group. The oracle user home directory is /oracle, which is a shared file system.
2. Varyon all shared volume groups.

3. Mount all shared file systems.
4. Copy the following files from the production server to the backup server:

```
/etc/pw-syscall
/etc/ora_kstat
/etc/loadext
/etc/oratab
```

5. Edit the /etc/inittab file and add the following line. Due to variations in AIX versions and Oracle versions, the entry might be different. You should copy the lines from /etc/inittab file on the production server.

```
orapw:2:wait:/etc/loadext -l /etc/pw-syscall
orakstat:2:wait:/etc/loadext -l /etc/ora_kstat
```

6. The entry you have just added to the /etc/inittab file is only activated at the next system reboot. If you do not want to reboot the backup system, execute the following commands:

```
/etc/loadext -l /etc/pw-syscall
/etc/loadext -l /etc/ora_kstat
```

7. Log in as oracle. At this point, the Oracle environment variable is already initialized because the user profile of the oracle ID is on the shared disk and is accessible by a user in the backup server.
8. Start Oracle.
9. Start the Oracle Listener.
10. Test your database.

The Oracle database is now successfully configured on both the production and the backup servers. Create a script to synchronize the system security and data on a non-shared disk. Refer to Chapter 3, “Cluster user IDs and user environments” on page 49 for more information about security synchronization.

7.4 Preparing an application server

An application server is defined in an HACMP resource with start and stop scripts. The start script and the stop script control the application initialization and shutdown.

7.4.1 Handling application startup errors

The following is the sequence of a cluster startup process:

1. Configure the service IP address if IP Address Takeover (IPAT) is configured.
2. Varyon all shared volume groups.
3. Mount all shared file systems.
4. Start the application server.

Once the application server event starts, the cluster startup process completes. The Cluster Manager does not evaluate the return code of the application server startup script, which means if the application fails to start, it does not affect the whole cluster startup process.

Therefore, we recommend that the startup scripts should be written to handle the return code of the application startup so that any failure in the application startup can be detected immediately.

7.4.2 Handling an application shutdown

Unlike the cluster startup process, an application server activation is the first event in the shutdown sequence. The application stop script is executed to stop the application, thus, releasing all the shared resources.

In a normal node failure, there is no shutdown sequence. Instead, the backup server performs a resource takeover as soon as it detects a node failure.

However, when performing a “graceful shutdown with takeover” of a cluster node, the application is stopped when there are users on the system. If the application shutdown fails, the shared resources are not released. Therefore, the cluster stop sequence fails, and the node takeover does not occur.

In order to avoid graceful takeover failure, a stop script must be able to shut down the application when there are active users. Most application provide options to shut down the application when there are active users.

7.4.2.1 A sample shutdown script modification

Let’s use Oracle software as an example. Oracle provides a startup and shutdown script to automate the process during system boot and system shutdown. By default, the shutdown script, dbshut, executes a shutdown command. If there are active users, the command hangs because it cannot perform a database shutdown.

In order to ensure that the shutdown is successful, the dbshut script must be modified as follows.

Replace the following lines:

```
connect internal
shutdown
EOF
```

With the following lines:

```
connect internal
shutdown immediate
EOF
```

The `shutdown immediate` command shuts down Oracle without waiting for users to close their sessions.

Chapter 8. HACMP and SAP R/3

SAP R/3 is an HACMP cluster environment that allows SAP R/3 information to be highly available. Since there is a growing demand to put SAP R/3 in an HACMP cluster, in this chapter, we explain how to customize the HACMP configuration to work with the SAP R/3 unique environment.

SAP R/3 can work with most RDBMSs, such as IBM DB2 UDB, Oracle, Informix, and Adabas. However, we use Oracle in our implementation example.

An SAP R/3 implementation may differ from one customer to another, but we use a basic configuration in an SAP R/3 environment. The following software levels are used as the base of our configuration:

- AIX 4.3.3
- HACMP/ES 4.3.1
- SAP R/3 45B
- Oracle 8.0.5.0

8.1 HACMP and SAP/R3 configuration

For the SAP and HACMP customization example, we chose to setup SAP R/3 and HACMP in a two-node cluster. One node is the DB server and the other node is the CI server. Each node is a backup to each other in case of a node failure.

Clients access the server through a token ring network, while the fast Ethernet network is designated as the server network. The application servers are not part of the HACMP cluster, but only HACMP clients.

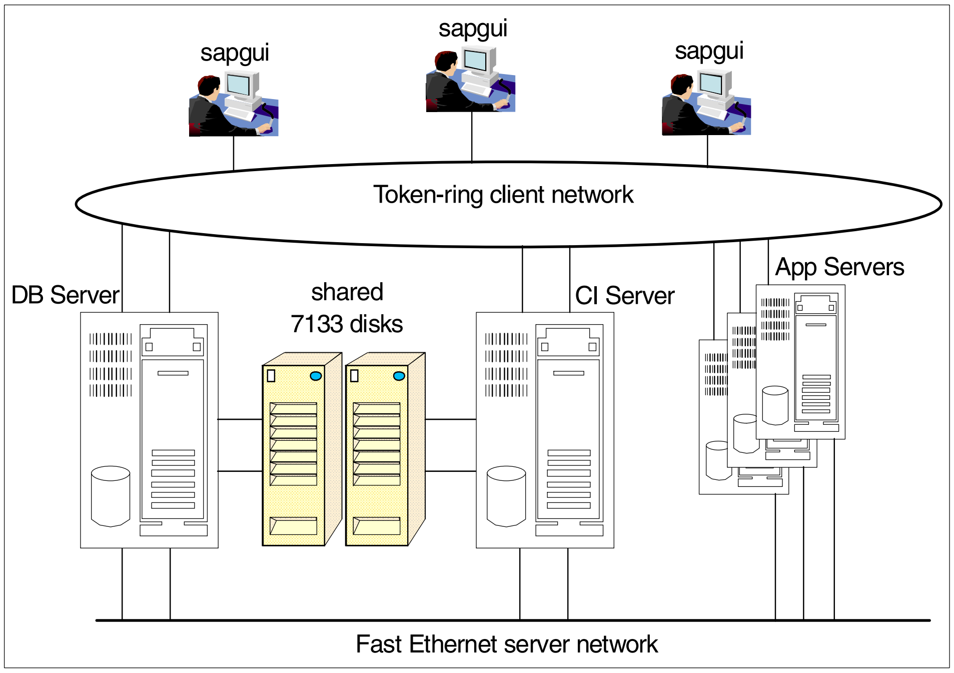


Figure 61. Cluster configuration for SAP R/3 implementation

The cluster configuration diagram shown in Figure 61 is the baseline of our cluster implementation.

8.2 Pre-installation tasks

Before the SAP R/3 can be integrated into an HACMP environment, some pre-installation steps should be taken to ensure that the SAP R/3 runs successfully in the high availability environment.

8.2.1 Security requirements

First of all, since the application moves from one server to another, it is important to make sure that the security of the cluster nodes allow the application to be started on the backup server, thus, allowing clients to regain access after a failover process. Consider the following:

- User and group definitions, password files, and home directory contents must be the same.
  - Be sure that SAP R/3 UID for <sid>adm, ora<sid>, and GID for sapsys and dba are the same on both systems.

- Time should be synchronized between cluster nodes.
- The /etc/passwd, /etc/group, and /etc/hosts files should be synchronized to all nodes.

8.2.2 SAP R/3 requirements

The following are SAP R/3 requirements:

- Paging space should be at least twice the size of the physical RAM.

  The paging requirements are for SAP R/3 only. Each RDBMS has a different paging space requirement.
- The following parameter should be set for root, <sid>adm, and ora <sid>:
  - Maximum file size: fsize=-1
- The following values in the operating system characteristics should be modified:
  - Maximum user process: maxuproc=500
  - Low water mark: minpout=24
  - High water mark: maxpout=33
- Asynchronous I/O (AIO) must be configured. The following are commands to configure AIO:

  ```
  mkdev -l aio0
  chdev -P -l aio0 -a autoconfig=available
  ```
- The home directory of the SAP administrator, <sid>adm, should have all the required files, and those files must be the same on both nodes.

8.2.3 SAP R/3 license requirement

HACMP requires SAP R/3 to be able to run on both cluster nodes so that the failover is successful. Therefore, a license key is required for each of the cluster nodes.

SAP R/3 supports the installation of multiple license keys since Release 3.0C. That enables us to install and setup the two valid license keys for the two nodes, the original license server and the takeover node, at the initial stage. Following that initial setup, our HACMP/6000 configuration is “license independent” until one of the nodes changes for any reason.

While we need an additional license key for an HACMP/6000 takeover node – the license key depends on SAP system ID (SID) and on the CPU number (Hardware)—SAP will give you one additional key for free. You need to add a remark to SAP “Fax Request” that you need an additional key for a HACMP/6000 or switch-over environment.

8.2.4 SAP R/3 installation path

The following documents the setup or installation path. If SAP is not installed, it can be installed either before or after you set up an HACMP cluster. It is important that shared file systems are created and ready for takeover. The failover cannot be properly tested until SAP is installed, since you cannot start the application resources.

If installing SAP before HACMP, you can save the effort of having to sort out the instance internal names in various profiles by using the target host name when asked for the server names during R3SETUP. Here, you are asked for the application server name and the target database name, which will be the client network. This properly sets up the network (tnsnames.ora, for example) and the server names in the SAP R/3 profiles. The startsap/stopsap and the environment profiles, which are dependent on the host name, still take on the value of the current host name. If you install before you have reset the host name, you need to add additional profiles.

8.3 Configuring an SAP R/3 environment

The environment variables for the SAP R/3 implementation are based on the cluster diagram shown in Figure 61 on page 110.

Table 2 shows the summary of the SAP R/3 environment that we used in our setup.

Table 2. SAP R/3 environment variables

Variable	Value
DB Server	ford
CI Server	dodge
SID	PRD

8.3.1 Network definition

The network adapter information is listed by node. All network information for node dodge is shown in Table 3 on page 113. There are two networks in this configuration. The main network is a token-ring network, which provides a communication link to the SAP clients. The second network is dedicated as a private network between cluster nodes and SAP application servers. The SAP NFS access uses the secondary network.

Table 3. Network adapters for node dodge

Adapter Label	IP Address	Network Type	Network Name	Function
Public network accessed by clients				
dodge	9.3.187.185	token ring	token1	service
dodge_boot	9.3.187.186	token ring	token1	boot
dodge_sby	10.1.1.183.	token ring	token1	standby
Private network for SAP R/3 servers communication				
dodge_en	10.1.2.185	Ethernet	ether1	service
dodge_en_boot	10.1.2.186	Ethernet	ether1	boot
dodge_en_sby	10.1.3.185	Ethernet	ether1	standby

A summary of the network adapter information for node ford is shown in Table 4.

Table 4. Network adapters for node ford

Adapter Label	IP Address	Network Type	Network Name	Function
Public network accessed by client				
ford	9.3.187.183	token ring	token1	service
ford_boot	9.3.187.184	token ring	token1	boot
ford_sby	10.1.1.183	token ring	token1	standby
Private network for SAP R/3 servers communication				
ford_en	10.1.2.183	Ethernet	ether1	service
ford_en_boot	10.1.2.184	Ethernet	ether1	boot
ford_en_sby	10.1.3.183	Ethernet	ether1	standby

8.3.2 SAP resource group

HACMP manages the SAP resource group shown in Table 5. The resources are configured as cascading resources. In summary, ford and dodge backup one another in case of a node failure.

Table 5. HACMP resource group

Resources	**CI Server**	**DB Server**
Participating nodes	dodge, ford	ford, dodge
Volume groups	nfsvg, civg	sapvg1, sapvg2
File systems	/sapexe /saptrans	/oracle/PRD/... /oracle/stage/...
HA application servers	sap_ci_nfs, sap_app_svr	sap_db_svr
Service IP	dodge, dodge_en	ford, ford_en

Information about the HA application servers is provided in Table 6.

Table 6. HACMP application server

HA Application Server	**Start Script**	**Stop Script**
sap_app_svr	/usr/sbin/cluster/saptools/start_r3_clients	/usr/sbin/cluster/saptools/stop_r3_clients
sap_db_svr	/etc/rc.startsap_prod_db	/etc/rc.stopsap_prod_db
sap_ci_nfs	/etc/rc.startsap_apl	/etc/rc.stopsap_apl

Note

The script file /etc/rc.startsap_prod_db was written for an environment where the CI server and DB server are running on one server. For use in the current environment, a modification was done on the script.

Line:

```
/usr/bin/su - $SAPADM "-c /home/$SAPADM/startsap_${DBSERVER}_$SAPNR ALL"
>> $OUT
```

Is replaced by:

```
/usr/bin/su - $SAPADM "-c /sapmnt/$SAPSID/exe/startdb " >> OU
```

8.3.3 Configuring NFS

We explore the setup of NFS file systems that are being used in the SAP R/3 environment to share the file systems with other SAP servers and clients.

8.3.3.1 SAP shared file systems

We begin by preparing the non-database file systems for takeover. These include /usr/sap/trans and /sapmnt, which are globally shared by all the servers and require NFS server takeover. In our configuration, these resources normally reside on the central instance server and are NFS exported from there (See Figure 62):

- /sapmnt/<SID> ---> /sapexe/<SID>
- /usr/sap/trans ----> /saptrans/<SID>

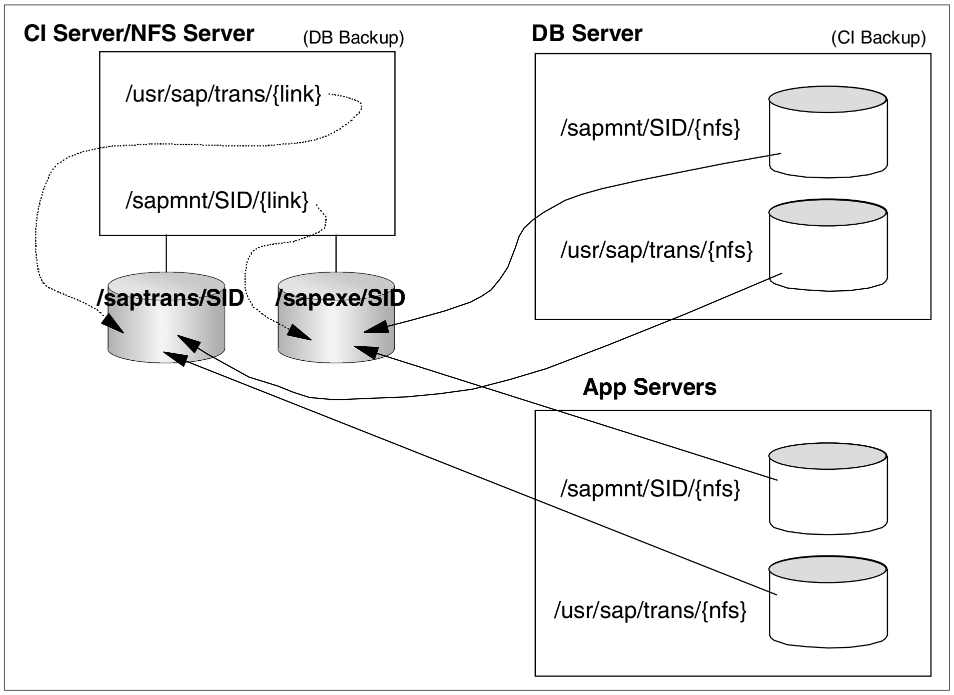


Figure 62. NFS file systems mounting before HACMP failover

In accordance with the ISICC HACMP Technical Brief recommendations, we first reorganize these file systems in order to avoid having to change HACMP scripts while still providing a smooth R3 takeover. In order to do this, we require a slightly different directory structure. The reason for the directory structure implemented is due to the way HACMP does a file system takeover.

These scripts do not differentiate between NFS mounted and local file systems.

After assuming the disks and file systems in a takeover, HACMP uses the mount command to check if the file systems obtained in the takeover are already mounted. If it finds a mount entry containing the name of this file system, whether the source or target of a mount, it does not mount the file system again. Therefore, if the file system was one that the failover node had mounted from its partner via NFS, and now has assumed control of, the assumed physical file system will not be mounted.

To avoid this scenario where the assumed file system is left unmounted, we take precautions that the file system that is exported, and the file system on which it is mounted, do not use the name of the physical file system that participates in the take over. See Table 7.

Table 7. NFS file systems relationship

Exported File System	**Physical File System**	**NFS Mount Point**
/sapmnt/<SID>	/sapexe	/sapmnt/<SID>
/saptrans/<SID>	/saptrans	/usr/sap/trans

The physical data is moved to a new location.

The original paths to the data are re-established using a symbolic link from the original location to the new location which is /usr/sap/trans/ and links to /saptrans/SID.

The CI server now exports the new file systems, /saptrans/SID and /sapexe/SID, which are mounted by the AppServer clients at the original path mount point. See Figure 63 on page 117.

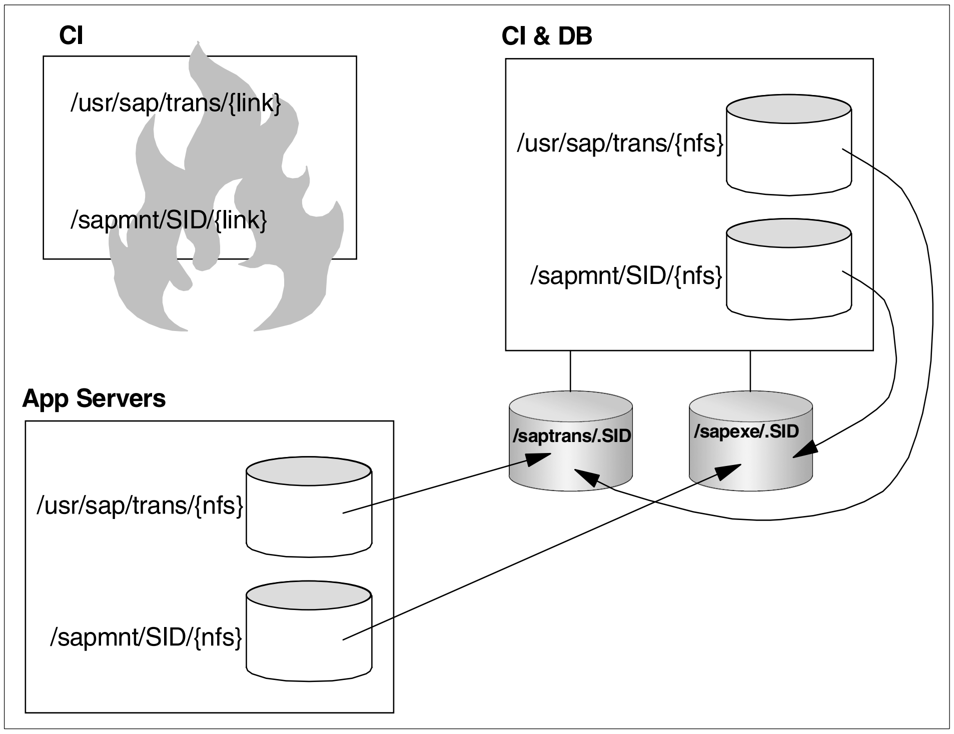


Figure 63. NFS file systems mounting after HACMP failover

The HACMP resources include the physical disk location of the data and the NFS server IP address. In the case of the CI server failing, as illustrated by the fire in Figure 63, the target disks are taken over by the backup, the CI servers service address is assumed, and the target file systems re-exported. In this failover configuration, the backup server performs as its own NFS server, as well as assuming this function for the other clients.

Important

- Remember to export /sapexe and /saptrans on the cluster nodes.
- Export and mount all NFS file systems on the private network.

8.3.3.2 Oracle client file system

Similar to the previous example for the shared file systems, we must make preparations for the takeover of the Oracle/SID file system, however, for different reasons in this case. With Oracle 8, the /oracle/SID file system exists on the clients as well as the DB server. In the case of the clients, only a subset of the file system exists containing the network connection information and the client libraries.

The following example configuration illustrated in Figure 64 shows one way of maintaining the local /oracle/SID in the normal production environment and the takeover of the DB servers /oracle/SID file system as the result of a failover.

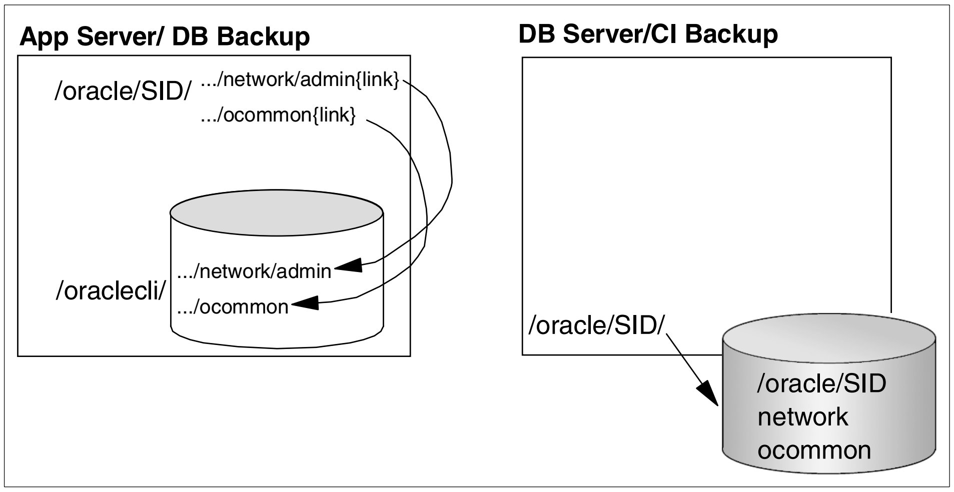


Figure 64. Oracle client file systems before HACMP failover

Links beneath /oracle/SID connect the necessary Oracle client directories into /oracle/SID. At failover, the DB server's /oracle/SID is mounted on top of the local mount point, covering over the links below. The fire in Figure 65 on page 119 illustrates that the DB server/CL Backup fails and the /oracle/SID disk is taken over by the App Server/DB Backup.

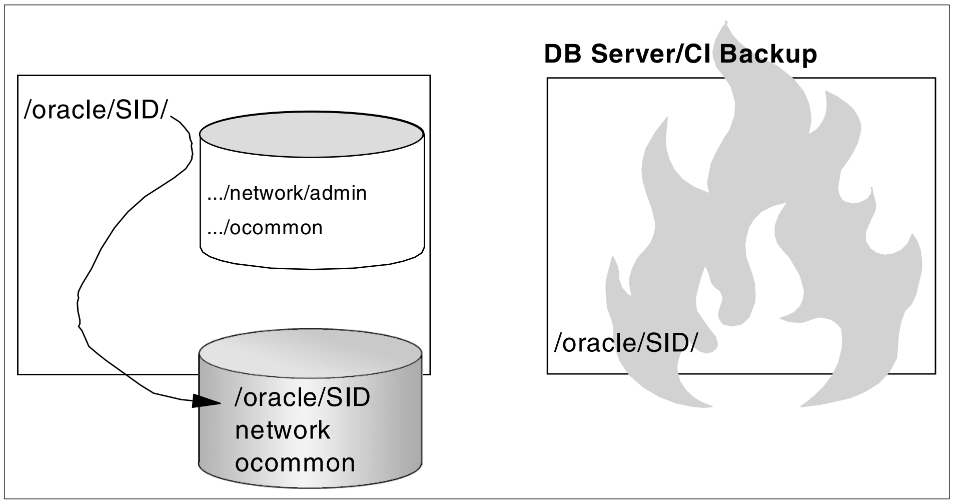


Figure 65. Oracle file systems after a failover

8.3.4 Sample NFS configuration

In this section, we go through an example of how to setup an NFS environment in our HACMP cluster. In the SAP R/3 environment, we do not use HACMP to manage NFS because of the limitation of NFS options that HACMP provides. We use the normal NFS management in AIX. The following sections go through the NFS setup for two file systems on a DB server.

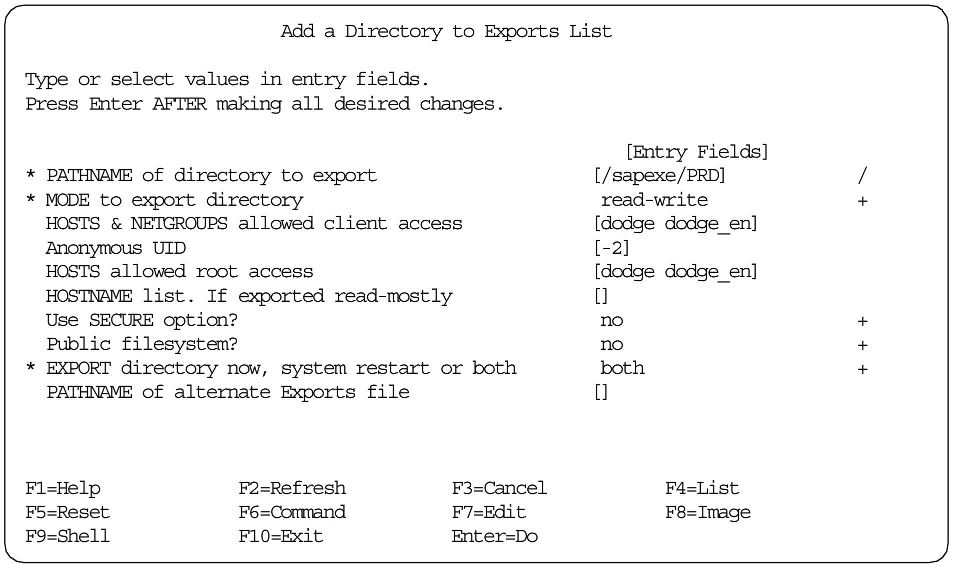
8.3.4.1 File system: /sapmnt/PRD

Complete the following steps:

1. Unmount the /sapmnt/PRD file system:
   ```
   umount /sapmnt/PRD
   ```
2. Rename /sapmnt/PRD to /sapexe:
   ```
   chfs -m /sapexe /sapmnt/PRD
   ```
3. Mount the new file system:
   ```
   mount /sapexe
   ```
4. Create a directory called /sapexe/PRD:
   ```
   mkdir /sapexe/PRD
   ```
5. Move the contents of /sapexe to /sapexe/PRD:
   ```
   mv /sapexe/* /sapexe/PRD/
   ```
6. Create a symbolic link from /sapmnt/PRD to /sapexe/PRD:

```
ln -s /sapexe/PRD /sapmnt/PRD
```

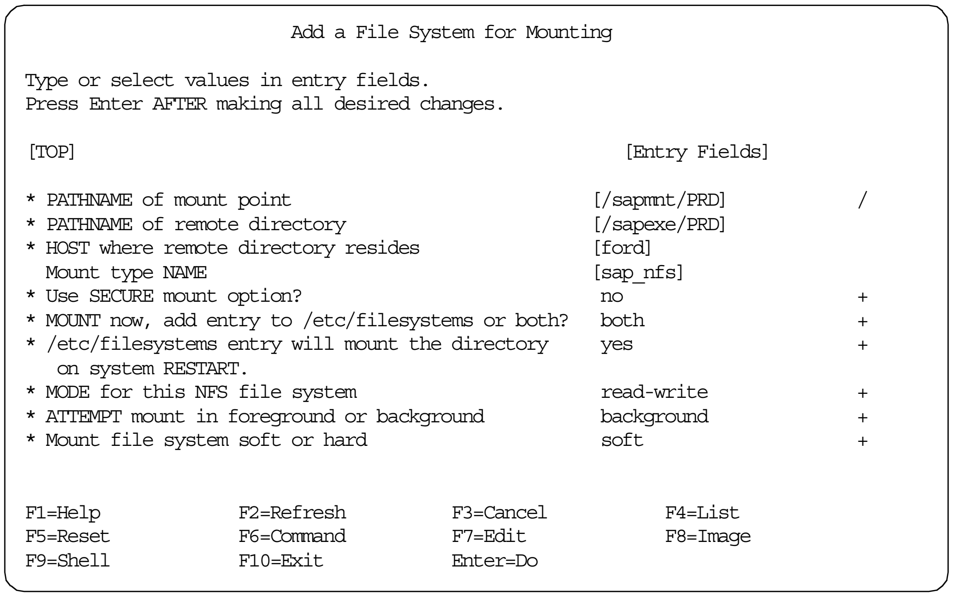
7. Export /sapexe/PRD using SMIT with the options shown in Figure 66.



```
                         Add a Directory to Exports List

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
* PATHNAME of directory to export                    [/sapexe/PRD]          /
* MODE to export directory                            read-write            +
  HOSTS & NETGROUPS allowed client access            [dodge dodge_en]
  Anonymous UID                                      [-2]
  HOSTS allowed root access                          [dodge dodge_en]
  HOSTNAME list. If exported read-mostly             []
  Use SECURE option?                                  no                    +
  Public filesystem?                                  no                    +
* EXPORT directory now, system restart or both        both                  +
  PATHNAME of alternate Exports file                 []



F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 66. Exporting the /sapexe/PRD file system

8. Mount /sapexe/PRD on the application server using SMIT with the options shown in Figure 67 on page 121.



```
                         Add a File System for Mounting

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                        [Entry Fields]

* PATHNAME of mount point                                 [/sapmnt/PRD]            /
* PATHNAME of remote directory                            [/sapexe/PRD]
* HOST where remote directory resides                     [ford]
  Mount type NAME                                         [sap_nfs]
* Use SECURE mount option?                                 no                      +
* MOUNT now, add entry to /etc/filesystems or both?        both                    +
* /etc/filesystems entry will mount the directory          yes                     +
   on system RESTART.
* MODE for this NFS file system                            read-write              +
* ATTEMPT mount in foreground or background                background              +
* Mount file system soft or hard                           soft                    +


F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 67. Mount an NFS file system as /sapmnt/PRD

Note

Some options of the SMIT screen shown in Figure 67 are deleted to show the required options only.

8.3.4.2 File system: /usr/sap/trans

Complete the following steps:

1. Unmount the /usr/sap/trans file system:

   ```
   umount /usr/sap/trans
   ```

2. Rename /usr/sap/trans to /saptrans:

   ```
   chfs -m /saptrans /usr/sap/trans
   ```

3. Mount the /saptrans file system:

   ```
   mount /saptrans
   ```

4. Create a new /saptrans/PRD directory:

   ```
   mkdir /saptrans/PRD
   ```

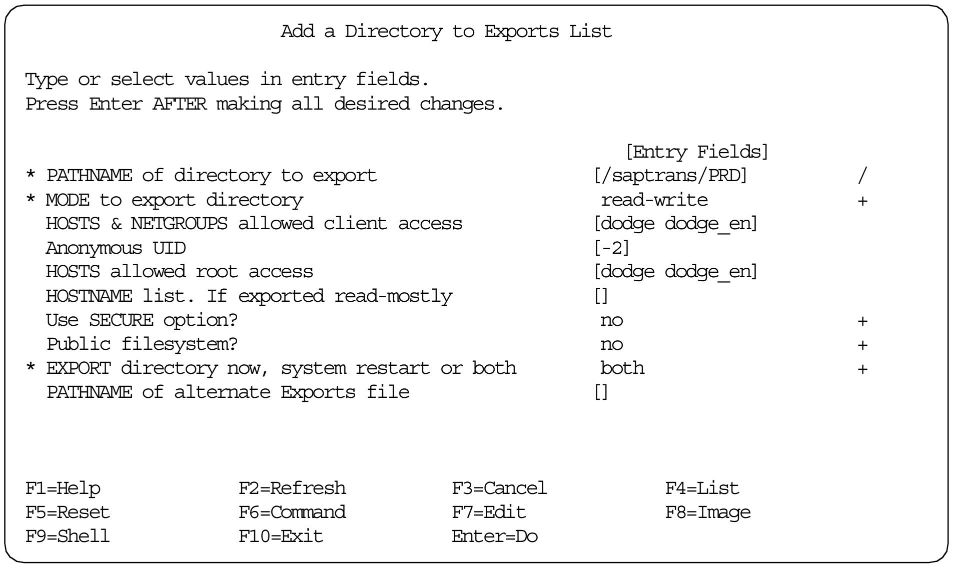
5. Move the contents of /saptrans to /saptrans/PRD:

   ```
   mv /saptrans/* /saptrans/PRD
   ```

6. Create a symbolic link from /usr/sap/trans to /saptrans/PRD:

```
ln -s /saptrans/PRD /usr/sap/trans
```

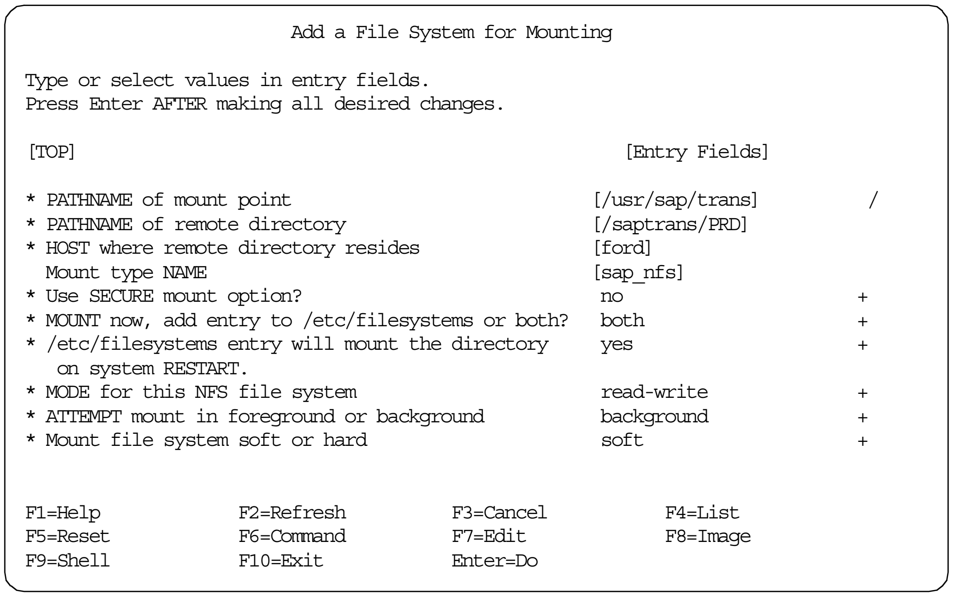
7. Export /saptrans/PRD using SMIT with the options shown in Figure 68.



```
                        Add a Directory to Exports List

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
* PATHNAME of directory to export                    [/saptrans/PRD]         /
* MODE to export directory                            read-write             +
  HOSTS & NETGROUPS allowed client access            [dodge dodge_en]
  Anonymous UID                                      [-2]
  HOSTS allowed root access                          [dodge dodge_en]
  HOSTNAME list. If exported read-mostly             []
  Use SECURE option?                                  no                     +
  Public filesystem?                                  no                     +
* EXPORT directory now, system restart or both        both                   +
  PATHNAME of alternate Exports file                 []



F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 68. Exporting the /saptrans/PRD file system

8. Mount /sapexe/PRD on the application server using SMIT with the options shown in Figure 69 on page 123.



```
                         Add a File System for Mounting

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                   [Entry Fields]

* PATHNAME of mount point                            [/usr/sap/trans]         /
* PATHNAME of remote directory                       [/saptrans/PRD]
* HOST where remote directory resides                [ford]
  Mount type NAME                                    [sap_nfs]
* Use SECURE mount option?                            no                     +
* MOUNT now, add entry to /etc/filesystems or both?   both                   +
* /etc/filesystems entry will mount the directory     yes                    +
   on system RESTART.
* MODE for this NFS file system                       read-write             +
* ATTEMPT mount in foreground or background           background             +
* Mount file system soft or hard                      soft                   +


F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 69. Mount an NFS file system as /saptrans//PRD

Note

Some options of the SMIT screen shown in Figure 69 are deleted to show the required options only.

8.3.5 Customizing the administrative user environment

The SAP environment should be configured to allow SAP processes to adapt to the HACMP cluster environment. The SAP environment variables determine how SAP applications behave in the primary node and the backup nodes. The environment setting includes customizing the profiles and environment variables.

8.3.5.1 Customizing SAP profiles

During an SAP R/3 installation, profiles are created to initialize a working environment for SAP R/3. Most profiles are dependent on the host on which it is executed. The following are SAP profiles that should be modified:

• Start profile: /sapmnt/PROFILE/profile/START_<Instance>_<hostname>
  - Copy START_DVEBMGS00_dodge to START_DVEBMGS00_ford.
  - Change all dodge to ford.

- Instance profile: /sapmnt/PRD/profile/PRD_<Instance>_<hostname>
  - Copy PRD_DVEBMGS00_dodge to PRD_DVEBMGS00_ford.
  - Add the parameter SAPLOCALHOST=dodge to PRD_DVEBMGS00_ford.

    Add it after the first lines where other environment variables are set.
- Default profile: /sapmnt/PRD/profile/DEFAULT.PFL
  - SAPDBHOST = ford
  - rdisp/mhost = dodge
  - rdisp/enqname = dodge_PRD_00
  - rdisp/vbname = dodge_PRD_00
  - rdisp/btcname = dodge_PRD_00
  - rdisp/sna_gateway = dodge

Note

SAPLOCALHOST is usually set to the R/3 server's host name at startup. If you set that parameter explicitly, R/3 believes it is running on the host specified. During a failover from dodge to ford, R/3 detects the host name ford and sets SAPLOCALHOST to ford. However, since the value of SAPLOCALHOST=dodge is explicitly defined in the profiles, R/3 believes it is running on dodge and uses dodge for all host name-dependent functions.

The following is a sequence of how R/3 sets the variable:

1. Directly from the system (default value)
2. From the default profile DEFAULT.PFL
3. Setting the parameter in the instance profile <SID>_<instance>_<hostname>

Entries in the default profile are IP labels only and are not host name dependent.

Entries in the START profile are name only.

There are no entries in the instance profile corresponding to the host name or any IP label, except our SAPLOCALHOST addition in the profile of the takeover node.

8.3.5.2 Customizing environment variables

Based on our sample configuration, the following are sample environment variables.

On the DB server:

```
/home/prdadm/startsap_<hostname>_<systemno>=startsap_ford_00
/home/prdadm/stopsap_<hostname>_<systemno>=stopsap_ford_00
```

The next two files are a copy of the previous two files, respectively:

```
/home/prdadm/startsap_<hostname>_<systemno>=startsap_dodge_00
/home/prdadm/stopsap_<hostname>_<systemno>=stopsap_dodge_00
```

However, you should change the following information in the two new files:

```
START_PROFILE=START_DVEMBGS00_dodge
SERVICE=CI
```

On CI server:

```
/home/prdadm/startsap_<hostname>_<systemno>=startsap_dodge_00
/home/prdadm/stopsap_<hostname>_<systemno>=stopsap_dodge_00
```

The next two files are a copy of the previous two files, respectively:

```
/home/prdadm/startsap_<hostname>_<systemno>=startsap_ford_00
/home/prdadm/stopsap_<hostname>_<systemno>=stopsap_ford_00
```

However, you should change the following information in the two new files:

```
START_PROFILE=START_DVEMBGS00_ford
SERVICE=CI
```

8.3.5.3 Customizing the Oracle environment

The default designated Oracle administrator user ID is ora<sid>, which, in our case, is oraprd. You have to make sure that the following files exist on both nodes. Since the /oracle file system is a shared file system, there is only one copy of each file. The following four files should be copied:

```
cp /oracle/PRD/.dbenv_ford.sh /oracle/PRD/.dbenv_dodge.sh
cp /oracle/PRD/.dbenv_ford.csh /oracle/PRD/.dbenv_dodge.csh
cp /oracle/PRD/.sapenv_ford.sh /oracle/PRD/.sapenv_dodge.sh
cp /oracle/PRD/.sapenv_ford.csh /oracle/PRD/.sapenv_dodge.csh
```

The following variable should be set to:

```
ORA_NLS33=/oracle/PRD/ocommon/nls/admin/data
```

For the default designated SAP administrator ID, prdadm (<sid>adm), you have to make sure that all the files exist in the home directory on both cluster nodes:

```
/home/prdadm/.dbenv_ford.sh
/home/prdadm/.dbenv_ford.csh
/home/prdadm /.sapenv_ford.sh
/home/prdadm /.sapenv_ford.csh
/home/prdadm/.dbenv_dodge.sh
/home/prdadm/.dbenv_dodge.csh
/home/prdadm /.sapenv_dodge.sh
/home/prdadm /.sapenv_dodge.csh
```

The following variable should be set to:

```
ORA_NLS33=/oracle/prd/ocommon/nls/admin/data
```

The variable can be found in the following Oracle administrator profiles:

```
/oracle/PRD/.dbenv_<hostname>.sh
/oracle/PRD/.dbenv_<hostname>.csh
```

Chapter 9. HACMP and TSM

Backup and recovery are mission-critical functions and, therefore, a target for HACMP installations. This chapter describes how to customize the TSM Server 3.7 and TSM Client 3.7 in a high availability environment. For the sake of simplicity, we address only the one-side takeover configuration, that is we have a cluster node, for example, ford, running the TSM server or TSM client and another cluster node, for example, dodge, running the TSM client. As ford fails, dodge takes over the TSM resources and brings back up the TSM server and possibly the ford TSM client.

9.1 The cluster environment

The cluster we use to install the TSM server is composed of two nodes: ford and dodge. The detailed configuration is shown in Section 1.7, “Web-based online planning sheets” on page 10 as an example of the online planning worksheet. We resume the configuration in Figure 70.

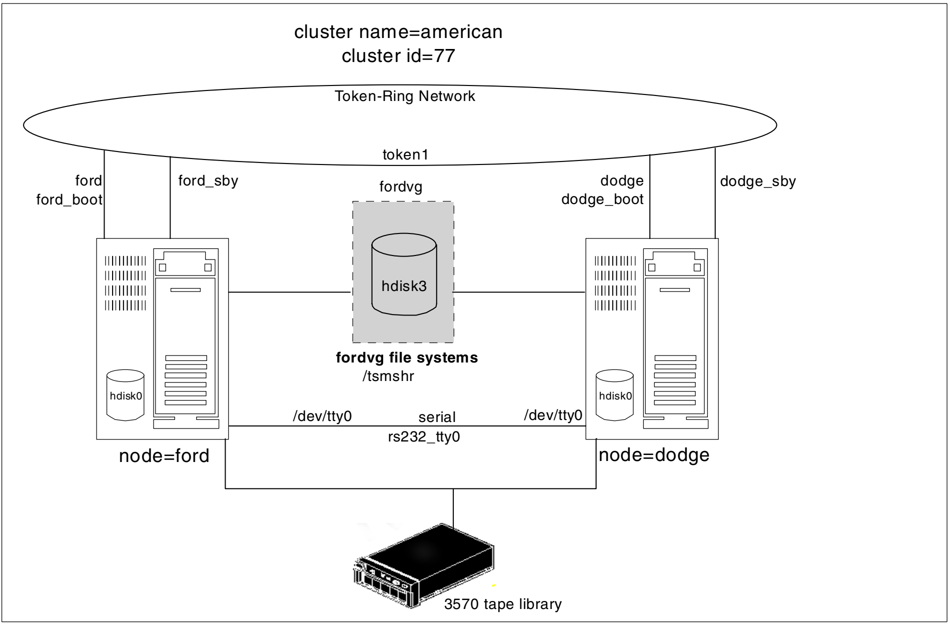


Figure 70. Cluster system for the TSM server

The nodes ford and dodge are 7013-550 models with 128 MB of RAM. The external storage is a 9333-500 SCSI system with three disks available. We are using only one disk for the TSM setup. The two nodes share a 3570 C02 tape library. The tape library is attached to an SCSI-2 DE Fast/Wide external adapter (FC 2416) with a 16-bit Differential Fast/Wide Y-Cable (FC 2426) to the node ford. The node dodge has also an SCSI-2 DE Fast/Wide external adapter with a 16-bit Differential Fast/Wide Y-Cable and the two Y-Cables are connected by a system-to-system cable (FC 2425).

The nodes ford and dodge have AIX 4.3.3 installed. The HACM/ES 4.3.1 is used for the present cluster. Finally, the TSM server and client version is 3.7.

9.2 Configuring a TSM server for a one-side takeover

This section describes how to install and configure TSM on a system in an HACMP cluster so that if one system fails, the TSM server is brought up on another system in the cluster. The primary node, for example, ford, runs the TSM server and we call it the production node; the secondary node, for example, dodge, is a standby node.

The installation and configuration steps may need some modifications in an environment that differs from ours, and we do not intend to supply a comprehensive treatment. We show an example to focus the common, basic requirements of the TSM server in a high availability cluster.

The procedure we follow is:

1. Define the shared volume group(s), the logical volume, and the file systems that will accommodate the TSM configuration files, database, recovery log, and storage pools. The *Tivoli Storage Manager for AIX Administrator's Guide,* GC35-0368*,* aids in determining how much disk space to allocate for each purpose. We use the sole shared file system /tsmshr.

Note

Integrity and performances are critical issues. You have to plan carefully how to protect your data and how to improve the I/O operation for the TSM database and storage pools. To provide the maximum availability, the shared volume group must be mirrored. To enhance performance, the I/O must be spread to avoid reading or writing bottlenecks. The access to the TSM files is mainly for updates, not for reads.

2. Configure HACMP such that the production node ford owns the TSM resources (shared file systems) and the standby node dodge has them as takeover resources (you can see the section Web-based online planning in

Chapter 1, “New functions in HACMP/ES 4.3.1” on page 1 for details about the HACMP configuration we implemented).

3. Configure HACMP so that the standby node takes over the production node IP address(es), which the TSM clients use to connect to the TSM server in the event of a node failure.

4. On both nodes, make a directory to download the executable and the script supplied in the associated material to this redbook at the following site:

   ```
   www.itso.ibm.com
   ```

   We create the directory /usr/es/sbin/cluster/local/tsmbin where we put the files:

   - start_tsm.sh, stop_tsm.sh, and verdev, which are kshell scripts
   - checkdev opendev scsitest and scsireset, which are executable

   Change the file permission:

   ```
   ford::root:/>cd /usr/es/sbin/cluster/local/tsmbin
   ford::root:/usr/es/sbin/cluster/local/tsmbin>chmod 755 *
   ```

5. Install the TSM server on both nodes following the instructions in *Tivoli Storage Manager for AIX Quick Start,* GC35-0367. The main TSM file components are:

   - The server executable and configuration files in the directory /usr/tivoli/tsm/server/bin
   - The client executable and configuration files in the directory /usr/tivoli/tsm/client/ba/bin
   - The api executable and configuration files in the directory /usr/tivoli/tsm/client/api/bin

6. Supply the basic options for administrative client access in the files dsm.opt and dsm.sys in the /usr/tivoli/tsm/client/ba/bin/ directory on the production node ford as shown in Figure 71.

```
ford::root:/usr/tivoli/tsm/client/ba/bin>more dsm.opt
Servername TSM *       A server name defined in the dsm.sys file

ford::root:/usr/tivoli/tsm/client/ba/bin>more dsm.sys
SErvername  TSM
   *COMMmethod         TCPip
   *TCPPort            1500
   COMMmethod         SHare
   TCPPort            1510
   TCPServeraddress   loopback
   PASSWORAccess      generate
```

Figure 71. The options in the dsm.sys and dsm.opt files

We show only the minimal options, but the option files should also contain the performance parameters as you can see in *Tivoli Storage Manager for UNIX, Using the Backup-Archive*, SH26-4105.

7. Verify the installation and start the server. Go to the /usr/tivoli/tsm/server/bin directory and check that the following files exist:
   - db.dsm (17 MB) and log.dsm (9 MB) are the default volumes for the TSM database and recovery log.
   - archive.dsm (8 MB) and backup.dsm (8MB) are the default volumes for the default disk storage pool.
   - dsmserv.dsk is the file listing the volumes used by the TSM database and recovery log.
   - dsmserv.opt is the TSM server option file.

   Sometimes, the /usr is not large enough to allocate the database and recovery log volumes. The installation finishes without warning, but you cannot start the server. In this case, a rescue procedure is needed to complete the installation or the installation has to be started again from the beginning after you add the needed space in the /usr file system.

   Then, you can start the server in the current directory (the TSM server can start only in the directory containing the dsmserv.dsk file).

8. Prepare for the high availability configuration of the TSM server. These steps allow you to configure your server to be brought back up on the standby node when the production node fails:
   - Start the cluster on the production node.
   - Connect to the server as administrative client via the command line or administrative Web interface (URL <present server IP address>:1580). Create a new database and recovery log volumes in a shared file

system (/tsmsshr in our lab environment) and delete the default database and recovery log volumes (be sure that the file system /tsmshr is mounted) as shown in Figure 72 on page 131.

```
ford::root:/>dsmadmc
Tivoli Storage Manager
Command Line Administrative Interface - Version 3, Release 7, Level 0.0
(C) Copyright IBM Corporation, 1990, 1999, All Rights Reserved.

Enter your user id:  admin

Enter your password:

Session established with server TSM: AIX-RS/6000
  Server Version 3, Release 7, Level 0.0
  Server date/time: 02/03/00   12:06:49  Last access: 02/03/00   12:06:25


tsm: TSM>def dbv /tsmshr/db.dsm formats=17
ANR2491I Volume Creation Process starting for /tsmshr/db.dsm, Process Id 3.
tsm: TSM>def logv /tsmshr/log.dsm formats=9
ANR2491I Volume Creation Process starting for /tsmshr/log.dsm, Process Id 4.
tsm: TSM>del dbv  /usr/tivoli/tsm/server/bin/db.dsm
ANR2243I Database volume /usr/tivoli/tsm/server/bin/db.dsm deleted.
tsm: TSM>del logv /usr/tivoli/tsm/server/bin/log.dsm
ANR2263I Recovery log volume /usr/tivoli/tsm/server/bin/log.dsm deleted.
```

Figure 72. The new volumes for the TSM database and recovery logs

The same must be performed with the disk storage pool volumes as shown in Figure 73 on page 132.

```
ford::root:/usr/es/sbin/cluster/local>dsmadmc
Tivoli Storage Manager
Command Line Administrative Interface - Version 3, Release 7, Level 0.0
(C) Copyright IBM Corporation, 1990, 1999, All Rights Reserved.

Enter your user id:  admin

Enter your password:

Session established with server TSM: AIX-RS/6000
  Server Version 3, Release 7, Level 0.0
  Server date/time: 02/03/00   17:11:22  Last access: 02/03/00   17:07:43


tsm: TSM>def v archivepool /tsmshr/archive.dsm formats=9
ANR2491I Volume Creation Process starting for /tsmshr/archive.dsm, Process Id
8.
tsm: TSM>def v backuppool /tsmshr/backup.dsm formats=9
ANR2491I Volume Creation Process starting for /tsmshr/backup.dsm, Process Id
9.
tsm: TSM>del v /usr/tivoli/tsm/server/bin/archive.dsm
ANR2220W This command will delete volume /usr/tivoli/tsm/server/bin/archive.dsm
from its storage pool after verifying that the volume contains no data.

Do you wish to proceed? (Yes/No) y
tsm: TSM>del v /usr/tivoli/tsm/server/bin/backup.dsm
ANR2220W This command will delete volume /usr/tivoli/tsm/server/bin/backup.dsm
from its storage pool after verifying that the volume contains no data.

Do you wish to proceed? (Yes/No) y
```

Figure 73. New disk storage pool volumes

- Stop the TSM server on the production node ford.
- Move the files dsmserv.dsk and dsmserv.opt from the /usr/tivoli/tsm/server/bin directory to the shared file system you have chosen. We issue:

  ```
  ford::root:/>mv /usr/tivoli/tsm/server/bin/dsmserv.opt /tsmshr
  ford::root:/>mv /usr/tivoli/tsm/server/bin/dsmserv.dsk /tsmshr
  ```

- From now on, you have to set environment variables to start the TSM server (see below and the *Tivoli Storage Manager for AIX Quick Start,* GC35-0367). The variables can be included in the /.profile or you can use the script start_tsm.sh you downloaded from:

  ```
  www.itso.ibm.com
  ```

  For future use, we move start_tsm.sh and stop_tsm.sh one level up. These scripts are going to be the start and stop scripts of the application server tsmapp we define in the HACMP configuration. See Figure 74 on page 133.

```
ford::root:/>cd /usr/es/sbin/cluster/local
ford::root:/usr/es/sbin/cluster/local>cp tsmbin/start_tsm.sh start_tsm.sh
ford::root:/usr/es/sbin/cluster/local>chmod 755 start_tsm.sh
ford::root:/usr/es/sbin/cluster/local>cp tsmbin/stop_tsm.sh stop_tsm.sh
ford::root:/usr/es/sbin/cluster/local>chmod 755 stop_tsm.sh
```

Figure 74. The start and stop scripts are moved

- Edit the start script /usr/es/sbin/cluster/start_tsm.sh as shown in Figure 75, Figure 76 on page 134, and Figure 77 on page 135.

```
#!/bin/ksh
################################################################################
#                                                                              #
# Shell script to start an ADSM/6000 server, after first making sure required  #
# offline storage devices are available.                                       #
#                                                                              #
# Please note commentary below indicating two places where this shell script   #
# may need to be modified in order to tailor it for your environment.          #
#                                                                              #
# Author: Steve Pittman                                                        #
#         IBM RS/6000 Marketing Specialist                                     #
#         (510)277-5080                                                        #
#                                                                              #
# Date: 12/6/94                                                                #
#                                                                              #
################################################################################
#
# Debug level
#
set -x
#
# Get file name of shell script
scrname=start_tsm.sh
# Get path to directory where shell script was found
bindir=/usr/es/sbin/cluster/local/tsmbin
```

Figure 75. The shell script start_tsm.sh: Part 1 of 3

```
# Define function to verify that offline storage device is available
VerifyDevice ()
{
        $bindir/verdev $1 &
        device[i]=$1
        process[i]=$!
        i=i+1
}
# Turn on ksh job monitor mode
set -m
#
echo "Verifying that offline storage devices are available..."
integer i=0
################################################################################
#                                                                              #
# Insert a VerifyDevice command below for every offline storage device used by #
# this ADSM server and shared by more than one system in the HACMP/6000        #
# cluster.                                                                     #
#                                                                              #
################################################################################
#VerifyDevice lb0
#VerifyDevice mt0
#VerifyDevice mt1
#
# Wait for all VerifyDevice processes to complete
#
wait
# Check return codes from all VerifyDevice (verdev) processes
integer allrc=0
tty=$(tty)
if [ $? != 0 ]
then tty=/dev/null
fi
jobs -ln | tee $tty | awk -v "encl=Done()" '{print $3, substr($4,length(encl),le
ngth($4)-length(encl))}' | while read jobproc rc
do
  if [ -z "$rc" ]
  then rc=0
  fi
  i=0
  while (( i < ${#process[*]} ))
  do
    if [ ${process[i]} = $jobproc ] ; then break ; fi
    i=i+1
  done
  if (( i >= ${#process[*]} ))
then
    echo "Process $jobproc not found in array!"
    exit 99
  fi
```

Figure 76. The shell script start_tsm.sh: Part 2 of 3

```
 if [ $rc != 0 ]
  then
    echo "Attempt to make offline storage device ${device[i]} available ended wi
th return code $rc!"
    allrc=$rc
  fi
done
if (( allrc ))
then exit $allrc
fi

echo "Starting ADSM now..."
################################################################################
#                                                                              #
# Update the cd command below to change to the directory that contains the     #
# dsmserv.dsk file and change the export commands to point to the dsmserv.opt  #
# file and /usr/lpp/adsmserv/bin directory for the ADSM/6000 server being      #
# started.  The export commands are currently set to the defaults.             #
#                                                                              #
################################################################################
cd /tsmshr
export DSMSERV_CONFIG=/tsmshr/dsmserv.opt
export DSMSERV_DIR=/usr/tivoli/tsm/server/bin
$DSMSERV_DIR/dsmserv quiet &
```

Figure 77. The shell script start_tsm.sh: Part 3 of 3

At line 18 of the shell script start_tsm.sh in Figure 75 on page 133, you supply the script name (for example, start_tsm.sh); at line 20, you enter the path where the script verdev is (for example, /usr/es/cluster/local/tsmbin). For the time being, lines 41, 42, and 43 are commented; we will come back to the devices as soon as we have defined them. Next, you must go at the end of the script and set the environment variables:

- DSMSERV_CONFIG that locates the file dsmserv.opt.
- DSMSERV_DIR that assigns the executable file path.

Eventually, you edit the script verdev as shown in Figure 78 on page 136, Figure 79 on page 137, and Figure 80 on page 138.

```
#!/bin/ksh
################################################################################
####
#                                                                              #
# Shell script to verify that an offline storage device is available and not
#
# SCSI RESERVEd by some other system.                                          #
#                                                                              #
# Author: Steve Pittman                                                        #
#         IBM RS/6000 Marketing Specialist                                     #
#         (510)277-5080                                                        #
#                                                                              #
# Date: 12/6/94                                                                #
#                                                                              #
################################################################################
####
# Get file name of shell script
scrname=verdev
# Get path to directory where shell script was found
bindir=/usr/es/sbin/cluster/local/tsmbin
# Return codes expected from scsitest command
timeout=78      # Request timed out
nomedia=102     # Media not loaded & ready in device
nowopen=122    # Device is already open by some other process on our system
reserved=124    # Device is reserved by some other system
DEVICE=$1
lsdev -C -l $DEVICE
STATUS=`lsdev -C -l $DEVICE -F status`
case "$STATUS" in
  Defined)
    #
    # Delete all devices at the same location
    #
    LOCATION=`lsdev -C -l $DEVICE -F location`
    lsdev -C | fgrep -e $LOCATION | fgrep -v $DEVICE | while read DUPDEV REST
    do
      lsdev -C -l $DUPDEV
      echo "`date +%H:%M:%S` Removing duplicate device $DUPDEV..."
      rmdev -l $DUPDEV -d
    done
    echo "`date +%H:%M:%S` Making defined device $DEVICE available..."
    mkdev -l $DEVICE
    rc=$?
    if [ $rc != 0 ]
    then
      echo "`date +%H:%M:%S` $scrname: mkdev command for device $DEVICE
generate
d unexpected return code of $rc!"
      exit $rc
    fi
;;
```

Figure 78. The shell script verdev: Part 1 of 3

```
 Available)
    #
    # See if device is reserved and, if so, break SCSI RESERVE
    #
    echo "`date +%H:%M:%S` Checking for SCSI RESERVE on device $DEVICE..."
    #
    # Note that we translate double quote (") and tab characters, thrown in by
    # the silly odmget command, to blanks.
    #
    PARENT=`odmget -q "name = $DEVICE" CuDv | tr '"     ' '  ' | awk '$1 == "par
ent" { print $3 }'`
    LOCATION=`lsdev -C -l $DEVICE -F location`
    for retryn in 1 2 3 4 5 6 7 8 9
    do
      $bindir/scsitest /dev/$PARENT $LOCATION
      rc=$?
      case $rc in
        # Ignore normal return, no media, and already open return codes
        0 | $nomedia | $nowopen)
          echo "`date +%H:%M:%S` Device $DEVICE not RESERVEd..."
          break
          ;;
        $timeout)
echo "`date +%H:%M:%S` $scrname: scsitest command timed out for device
 $DEVICE - retry $retryn ..."
          continue
          ;;
        $reserved)
          echo "`date +%H:%M:%S` Breaking SCSI RESERVE on device $DEVICE..."
          $bindir/scsireset /dev/$PARENT $LOCATION
          rc=$?
          if [ $rc != 0 ]
          then
            echo "`date +%H:%M:%S` $scrname: scsireset command for device $DEVIC
E generated unexpected return code of $rc!"
            exit $rc
          else
            echo "`date +%H:%M:%S` Waiting for break to complete on device $DEVI
CE..."
            $bindir/checkdev /dev/$DEVICE
            rc=$?
            case $rc in
              0) ;;
              46)
                echo "`date +%H:%M:%S` $scrname: 'Device not ready' message from
 checkdev is normal."
echo "`date +%H:%M:%S` $scrname: 'Device not ready' means media
is not mounted and ready on device $DEVICE."
                ;;
              # I/O error or device busy indicates SCSI RESERVE still set
              5 | 16)
                echo "`date +%H:%M:%S` $scrname: scsireset command failed to bre
ak SCSI RESERVE on device $DEVICE!"
                exit $rc
                ;;
```

Figure 79. The shell script verdev: Part 2 of 3

```
 *)
               echo "`date +%H:%M:%S` $scrname: checkdev command for device $DE
VICE generated unexpected return code of $rc!"
               exit $rc
               ;;
             esac
           fi
           break
           ;;
         *)
           echo "`date +%H:%M:%S` $scrname: scsitest command for device $DEVICE g
enerated unexpected return code of $rc!"
           exit $rc
           ;;
       esac
     done
     ;;
  *)
     echo "`date +%H:%M:%S` $scrname: lsdev command for device $DEVICE returned u
nexpected status of $STATUS!"
     exit 98
     ;;
esac
```

Figure 80. The shell script verdev: Part 3 of 3

At line 15 of the script verdev shown in Figure 78 on page 136, you supply the script name (that is, verdev), and at line 15, you enter the path for the executables scsireset, scsitest, chevdev, and opendev (for example, /usr/es/cluster/bin/local/tsmbin). You can also check the script stop_tsm.sh as shown in Figure 81 on page 139.

```
#!/bin/ksh
################################################################################
# Shell script to stop an ADSM/6000 server.                                    #
# Please note that changes must be made to the dsmadmc command below in order  #
# to tailor it for your environment:                                           #
#    1. Set -servername= to the ADSM/6000 server name on the SErvername option #
#       in the /usr/lpp/adsm/bin/dsm.sys file.                                 #
#    2. Set -id= and -password= to an ADSM userid that has been granted        #
#       operator authority, as described in the section titled "Step 6:       #
#       Registering Additional Administrators" in the installation manual.     #
# Author: Steve Pittman                                                        #
# Date: 12/6/94                                                                #
################################################################################
#
# Debug level
#
set -x
#
# Set seconds to sleep.
secs=5
#
echo "Stopping ADSM now..."
/usr/tivoli/tsm/client/ba/bin/dsmadmc -servername=TSM -id=admin -password=admin
-noconfirm << EOF
halt
EOF
echo "Sleeping $secs seconds to allow ADSM to stop..."
sleep $secs
```

Figure 81. The shell script stop_tsm.sh

The parameter secs, id, and password must be customized as the script header suggests.

- Test the start and stop scripts. If they fail, check the environment variables setting in the start script and the path for the halt command in the stop script.
- Define the application server (for example, tsmapp) that starts and stops the TSM server. Assign the start and stop scripts to the application server (for example, /usr/es/sbin/cluster/local/tsm_start.sh and /usr/es/sbin/cluster/local/tsm_stop.sh). Then, place the application server in a cascading resource group (in our example, fordrg) that has the production node and the standby node as participating nodes.
- Now, copy the start script (start_tsm.sh) and the stop script (stop_tsm.sh) to the standby node and synchronize the cluster.
- You are ready for a takeover test.

9. The previous long step completes the basic setup of the TSM server in a high availability environment. Now, you can continue with the TSM server

customization. The next section gives some hints about how to define the tapes and the tape libraries.

9.3 The highly available configuration of backup devices

The backup devices managed by TSM fill a long list as you can see at:

```
www.tivoli.com/tsm
```

Therefore, this section does not cover the configuration of any single device model. We present the basic rules to have a shared device in the highly available environment.

The example we supply at the beginning of the chapter tells you how to make a SCSI library available to both the cluster nodes. The library is connected to the first node via the short leg of a Y-cable, then the long leg of the Y-cable is attached to the long leg of a similar Y-cable on the second node. Eventually, the short leg of the second node Y-cable is terminated.

When the hardware configuration is ready, and you have installed the proper device driver, you can configure the devices to the operating system. Remember that the SCSI adapter has, by default, the ID number 7 and you are putting two adapters on the same SCSI chain. Therefore, you must change at least one of the SCSI IDs, issue the command `smit device`, and go through the SMIT panel **Devices -> SCSI Adapter -> Change / Show Characteristics of a SCSI Adapter**, select the proper adapter, and fill the Adapter card SCSI ID field with the new value. Issue the `cfgmgr` command on both nodes. Then, check the device logical names with the `lsdev -Cc tape` command. Each device must have the same name on both nodes. You may have to define dummy devices on one of the nodes to accomplish this. In this case, issue `smit device` and go through the SMIT panels to define a device. Choose an unused SCSI address for the device. Just before you press **Enter** to define the device, press **F6** to display the command SMIT is about to execute. Then, exit SMIT and go back to the command line and enter the same command with an extra `-d` flag.

Now, you can define the offline storage devices to the TSM server and you can edit the start script start_tsm.sh to add the device names (line 41). The verdev script checks the device availability and removes the SCSI locks that the system cannot clean up in a failure event. Next, you have to test a takeover to be sure that the TSM server also configures the devices when it run in the standby node.

9.4 Configuring a TSM client for a one-side takeover

The TSM server is always a TSM client; therefore, it is mandatory to go through the configuration of the TSM client in a high availability environment. Moreover, the cluster nodes can be merely TSM clients, but we want the resources to be backuped also when they are taken over by standby node. You can define an HACMP application server, for example, tsmclientapp, that manages the TSM client. We associate /usr/es/sbin/cluster/local/start_tsm_client.sh and /usr/es/sbin/cluster/local/stop_tsm_client.sh as start and stop script to the application server (these script can be downloaded in www.itso.ibm.com). When the production node is both TSM server and TSM client you can merge the application server tsmapp and tsmclientapp and their scripts.

We distinguish two main scenarios.

9.4.1 TSM client: The host name is a takeover resource

The TSM server registers the TSM client via the client host name (as default). If the production node host name is taken over as the production node fails, the TSM client must be validated again to the server as it starts to run on the standby node. The TSM validation procedure encrypts the client password in the /etc/security/adsm/TSM file. Therefore, you have to copy this file from the production node to the standby node.
Moreover, the options the TSM client needs to run on the standby nodes could differ from the options it has on the production node.
Then, we can focus on two subtopics in the TSM client high availability: option files and security. The start script start_tsm_client.sh in Figure 82 on page 142 takes care of both items.

```
#!/bin/ksh
################################################################################
#                                                                              #
# Shell script to configure TSM client after a takeover event.                #
#                                                                              #
# Please note commentary below indicating two places where this shell script   #
# may need to be modified in order to tailor it for your environment.          #
#                                                                              #
# Date: 4/2/2000                                                               #
#                                                                              #
################################################################################
#
# Debugging level
set -x
#
# Get the production and standby node names
#
productionnode=ford
standbynode=dodge
#
# Option file stanza
#
if [[ `/usr/es/sbin/cluster/utilities/get_local_nodename` = $standbynode ]]
then
cp -p /usr/tivoli/tsm/client/ba/bin/dsm.opt /usr/tivoli/tsm/client/ba/bin/dsm.op
t.$standbynode
cp -p /usr/tivoli/tsm/client/ba/bin/dsm.opt.$productionnode /usr/tivoli/tsm/clie
nt/ba/bin/dsm.opt
fi
#
# Security stanza: choose one security method
#
if [[ `/usr/es/sbin/cluster/utilities/get_local_nodename` = $standbynode ]]
then
#
# first security option: authorization is turend off till the next
# system administartor action
#
#
#sleep 30 #to be sure the TSM runs
#dsmadmc -id=admin -password=admin set auth off
#
#second security option; to use this option you must check that the file
# /etc/security/adsm/TSM.$productionnode exists on node $standbynode and
# that it is updated to the last $productionnode TSM client password
#
cp -p /etc/security/adsm/TSM /etc/security/adsm/TSM.$standbynode
cp -p /etc/security/adsm/TSM.$productionnode /etc/security/adsm/TSM
fi
```

Figure 82. The shell script start_tsm_client.sh

9.4.1.1 The option file issue

Since the standby node is usually a TSM client, too, the client option files dsm.sys and dsm.opt must be changed when a takeover event takes place. Next, we copy the production node client option file to the standby node:

```
ford::root:/usr/tivoli/tsm/client/ba/bin>rcp -p dsm.sys \
> dodge:$PWD/dsm.sys.ford
```

The dsm.sys.ford file is meant to replace the original standby node file dsm.sys in the failure event as you can see in the script start_tsm_client.sh in Figure 82 on page 142.

9.4.1.2 The security issue

The TSM client of the production node starts to run on the standby node; it must be validated by the TSM server. The client validation can be managed by the system administrator as you chose the first option in the start script as shown in Figure 82 on page 142. Otherwise, you can choose the second security option. Then, the TSM client encrypted password file must be copied from the production node to the standby node anytime the client password changes, for example:

```
ford::root:/>rcp -p /etc/security/adsm/TSM
dodge:/etc/security/adsm/TSM.ford
```

At the same time, you must supply a backup copy of the original encrypted password file on node dodge:

```
dodge::root:/>cp -p /etc/security/adsm/TSM /etc/security/adsm/TSM.dodge
```

This file is used during the ford reintegration and allows the TSM client on dodge to be validated as the TSM server starts on node ford.
When using this method, keep in mind that the change of the password on the production node must be always propagated to the standby node (coping the encrypted TSM password file). Forgetting this step results in an authorization failure for the TSM client after a takeover event. A manual intervention by the system administrator fixes this problem.

The stop script stop_tsm_client.sh is displayed in Figure 83 on page 144.

```
#!/bin/ksh
###############################################################################
#                                                                             #
# Shell script to configure TSM client after a rejoin event.                  #
#                                                                             #
# Please note commentary below indicating two places where this shell script  #
# may need to be modified in order to tailor it for your environment.         #
#                                                                             #
# Date: 4/2/2000                                                              #
#                                                                             #
###############################################################################
#
# Debugging level
set -x
#
# Get the production and standby node names
#
productionnode=ford
standbynode=dodge
#
# Option file stanza
#
if [[ `/usr/es/sbin/cluster/utilities/get_local_nodename` = $standbynode ]]
then
cp -p /usr/tivoli/tsm/client/ba/bin/dsm.opt.$standbynode /usr/tivoli/tsm/client/
ba/bin/dsm.opt
fi
#
# Security stanza: choose one security method
#
if [[ `/usr/es/sbin/cluster/utilities/get_local_nodename` = $standbynode ]]
then
#
# first security option: authorization is turend on till the next
# system administartor action
#
#dsmadmc -id=admin -password=admin set auth on
#
#second security option; to use this option you must check that the file
# /etc/security/adsm/TSM.ford exists on node dodge and that it is updated
# to the last ford TSM client password
#
cp -p /etc/security/adsm/TSM.$standbynode /etc/security/adsm/TSM
fi
```

Figure 83. The shell script stop_tsm_client.sh

This script recovers the original option file in the standby node as the production node rejoins the cluster as shown in Figure 83. The authorization is turned on and the original encrypted TSM password file is retrieved. You have to choose between the authorization method with the choice you made in the start script start_tsm_clienet.sh.

When you chose the security option that turns off and on the TSM authorization method, and the TSM client is running on the same node as the TSM server, you must be careful. In this case, the start and stop script must be merged in the start and stop scripts of the application server that manages the TSM server. The script start_tsm_client.sh must be appended to the script start_tsm.sh, whereas the script stop_tsm_client.sh must be inserted just above the stopping of the TSM server in the script stop_tsm.sh.

9.4.2 TSM client: The host name is not a takeover resource

This section explains the case where the standby node does not take over the production node host name. After the failure event, two TSM clients run on the standby node: the original TSM client, for example, dodge, and the former production node TSM client, for example, ford. To have the two clients on the same node, you must go through the following two steps:

1. Modify the file dsm.sys on the standby node dodge so that it has two stanzas as shown in Figure 84.

```
SErvername  TSM
   COMMmethod        TCPip
   TCPPort           1500
*   COMMmethod        SHare
*   TCPPort           1510
   TCPServeraddress    < TSM server addres>
   Passwordacces generate

SErvername  TSM_TAKEOVER
   Nodename ford
   COMMmethod        TCPip
   TCPPort           1500
*  COMMmethod         SHare
*  TCPPort           1510
   TCPServeraddress    < TSM server addres>
   Passwordacces generate
```

Figure 84. The two stanzas in the dsm.sys file

   The first one concerns the original standby TSM client, the second one instead concerns the TSM client formerly running on the standby node. If the TSM server is on the failing node as well, then in the second stanza, you can use the shared memory communication method.

2. Copy the production node option file dsm.sys on the standby node with a different name, for example, dsm.opt.ford:

```
ford::root:/usr/tivoli/tsm/client/ba/bin>rcp -p dsm.opt
dodge:$PWD/dsm.opt.ford
```

You edit the dsm.opt.ford file on the standby node and modify the Servername entry:

```
Servername TSM_TAKEOVER
```

The TSM client sections can start with the option file that matches the server name in the second stanza in the dsm.sys file as shown in Figure 84 on page 145.

Then, you use the start script start_tsm_client2.sh displayed in Figure 85 for the application server.

```
#!/bin/ksh
################################################################################
#                                                                              #
# Shell script to configure TSM client after a takeover event.                 #
#                                                                              #
# Please note commentary below indicating two places where this shell script   #
# may need to be modified in order to tailor it for your environment.          #
#                                                                              #
# Date: 4/2/2000                                                               #
#                                                                              #
################################################################################
#
# Debugging level
#
set -x
#
# Get the production and standby node names
#
productionnode=ford
standbynode=dodge
#
# Security stanza: choose one security method
#
if [[ `/usr/es/sbin/cluster/utilities/get_local_nodename` = $standbynode ]]
then
dsmadmc -id=admin -password=admin set auth off
fi
```

Figure 85. The shell script start_tsm_client2.sh

We decide to turn off the TSM authorization method because we want two TSM clients running in the same system and we want the client session to start in an unattended mode. The encryption TSM method is not going to fit this need as far as we know. The stop script stop_tsm_client2.sh of the application server shown in Figure 86 on page 147 then has only to turn on the authorization method as well.

```
#!/bin/ksh
###############################################################################
#                                                                             #
# Shell script to configure TSM client after a rejoin event.                  #
#                                                                             #
# Please note commentary below indicating two places where this shell script  #
# may need to be modified in order to tailor it for your environment.         #
#                                                                             #
# Date: 4/2/2000                                                              #
#                                                                             #
###############################################################################
#
# Debugging level
#
set -x
# Get the roduction and standby node names
#
productionnode=ford
standbynode=${standbynodename}
#
# Security stanza: choose one security method
#
if [[ `/usr/es/sbin/cluster/utilities/get_local_nodename` = $standbynode ]]
then
dsmadmc -id=admin -password=admin set auth on
fi
```

Figure 86. The shell script stop_tsm_client2.sh

When the two TSM clients run in standby node, you must start the TSM client sessions:

- The usual `dsmc` command starts the TSM client in the standby node (in our example, the client dodge).
- The following command starts the client sessions formerly running on the production node (in our example, ford):

  ```
  dodge::root:/>dsmc -optfile=/usr/tivoli/tsm/client/ba/bin/dsm.opt.ford
  ```

  In the case where the production node is also a TSM server, the start and stop scripts must be merged in the start and stop script of the application server that manages the TSM server.

9.5 The TSM API and Tivoli data protection for applications

The application data are saved by TSM through the API or connect agent, which interface the application backup tools with the TSM server. The variety of the agents prevents us from treating them in depth. Nevertheless, we can discuss some considerations.

The APIs have the same configuration files as the backup/archive (ba) client, and you can easily figure how to continue with the high availability API setup.

For Tivoli data protection for applications, you must go through the details of each one. The connection to the TSM server and the authorization method are based on the underlining TSM ba client as previously discussed. In order to complete the high availability of the Tivoli data protection for applications, you have to go further, creating the same environment on the production node and on the standby node. For example, the same user must perform the backup on both the cluster nodes; this user must set the same environment variables, and so forth.

9.6 Summary

In this chapter, we described the steps needed to configure the high availability TSM server and outlined the two main TSM client scenarios. The high availability TSM client is also the preliminary step for a high availability backup system for enterprise-critical applications.

We continue discussing TSM server availability in two main items:

- The configuration files (dsmserv.opt and dsmserv.dsk), the database, the recovery logs, and the disk storage pool volumes must stay in shared resources. The TSM environment variable DSMSERV_CONFIG must point to the customized locations of the configuration files. The variable DSMSERV_DIR allows you to start the server from the shared file system while keeping the TSM executable in the default installation position.
- The cluster nodes must access the offline storage devices, and a procedure has to clean up the SCSI locks in a failure event.

Chapter 10. HACMP and the disk subsystem

This chapter describes how to configure your serial storage architecture (SSA) disk subsystem for high availability in an HACMP environment. We discuss the new data protection features of the Advanced Serial RAID adapter and AIX 4.3.

The Advanced SSA Optical Extender and Enterprise Storage Server Peer-to-Peer Remote Copy break the limitations of an HACMP cluster. This technology enables you to separate the cluster nodes up to 100 kilometers. HACMP, with fiber-connected disks, can be used for real-site protection, and it is an alternative to HAGEO. In the second part of this chapter, we give some advice about how to configure your cluster and fiber-connected disks for full site protection.

10.1 SSA disks: Planning for availability

In an HACMP environment, you need to carefully design the cluster and its components so no device becomes a single point of failure. For example, it is not worthwhile to have a cluster if the data disks are not mirrored.

The entire SSA subsystem is designed for high availability. It supports hot-plugable disks, power supply, fan module, and even the data cables can be changed online. An SSA drawer can be connected with up to eight host systems, and an SSA loop can contain two adapter cards per host.

The following scenarios need to be considered to achieve data availability using an SSA disk subsystem:

Host failure	All SSA adapters and disk drawers enable multiple host configuration. If all loops connect to more than one hosts, a host failure can be tolerated together with HACMP. If a host is powered down, the bypass cards in the disk drawer close the loops, so other hosts can reach the disks.
Adapter failure	SSA supports two adapters in the same host in the same loop. In case of adapter failure, the SSA device driver reroutes the traffic automatically to the good adapter.

Drawer failure	The 7133 disk drawer has redundant, hot-plugable power supplies and fan modules. Also, the power supplies support using two different power sources.
Disk failure	Either AIX LVM mirroring or hardware RAID 1, 5, and 10 can be used to avoid disk failures. Both solutions support hot spare disks.
Cable or connector failure	In case of a cable or connector failure, the SSA device drivers reroute the data traffic. Only multiple cable failures can cause loss of availability of disks. The cables are also hot-plugable.

Figure 87 shows a highly available SSA disk configuration.

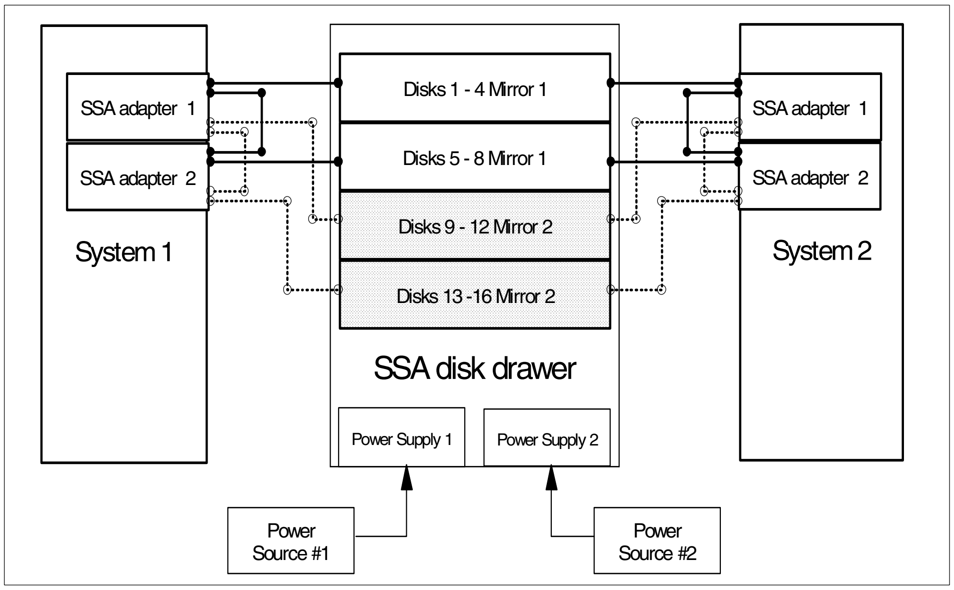


Figure 87. Highly available SSA disk configuration

In this two node configuration, we eliminate all possible single points of failure. We have two SSA adapters per host, each connected to both loops. The SSA device driver assigns the disks with a primary adapter card. If the primary disk controller fails, the SSA device driver routing algorithm reconfigures its preferred path to the disks, so all disks remain available. If a host is powered down, the SSA drawer bypass cards close the loops. If a disk drive fails, the SSA loop may be broken. The disk controllers reroute the traffic, so all disks can be reached from both hosts. The SSA drawer is connected to two separate power sources. This maintains the availability of the disks in case of power or UPS failure. And finally, all disks in the SSA drawer are mirrored.

10.1.1 AIX LVM mirroring and SSA hardware RAID options

The AIX Logical Volume Manager (LVM) provides commands to manipulate the storage disks. It enables the creation of volume groups including one or more physical hard disks. The logical volumes, which contain the file systems and other data structures, are highly configurable. There are a dozen of easily manageable variables that determine where and how the logical volume places out. These parameters define the physical disk names, the logical volume's location within a disk, the number of mirrors, and so on.

Together, AIX and SSA subsystems provide a wide variety of methods to improve data availability, or system performance. This includes software and hardware mirroring, striping, and their combinations. The following mirroring and RAID options are available:

- AIX LVM disk striping
- AIX LVM disk mirroring
- AIX LVM disk striping and mirroring
- Hardware RAID 0
- Hardware RAID 1
- Hardware RAID 5
- Hardware RAID 10 (0+1)

In the following sections, we describe the features and differences of the data storage methods.

10.1.1.1 AIX LVM disk striping

Disk striping provides faster data access by distributing the data across the disks in the volume group. Disk striping does not provide redundant information, so it is not the right choice for storing data in an HACMP cluster. Its read/write performance is outstanding on large, serial files.

10.1.1.2 AIX LVM disk mirroring

The Logical Volume Manager provides two- or three-way mirroring. The LVM mirrors the logical volumes but not disks. This allows mirroring for only the mission critical data and not the entire volume group. The LVM allows any number of disks in a volume group—it can mirror an odd number of disks. Since you can put different types of disks in the same volume group, you can mirror different types of disks, such as a mixture of SSA and SCSI disks. The LVM mirroring does not include an automatic replacing mechanism for failed disks. The SSA hot spare tool can be used only on mirrored SSA disks.

The LVM mirroring's read performance is good for short length operations, as long as data can read from any of the mirrored disks. The writes are performed on all disks, so it causes a higher I/O load on the system.

10.1.1.3 AIX LVM disk striping and mirroring

The simultaneous disk striping and mirroring was introduced in AIX 4.3.3. This data storage method combines the advantages of the LVM disk striping and mirroring. The data is striped across mirrored disks. It provides greater performance than the LVM disk mirroring and it is also redundant. It is a good choice for speed and availability.

10.1.1.4 RAID 0 in an Advanced SerialRAID adapter

Hardware RAID 0 is supported on the Advanced SerialRAID adapter (FC 6225 and 6230). RAID 0 stripes the data across the disks. It has an excellent read/write performance. A RAID 0 array can be built from 2 to 16 disks. All disks must be in the same loop. Because this storage policy does not provide redundant information, do not store your mission-critical data on a RAID 0 array. Also, RAID 0 configuration does not support multiple host attachment.

10.1.1.5 RAID 1 in an Advanced SerialRAID adapter

Hardware RAID 1 is simple data mirroring. It is the ideal storage mode where data availability is the key concern. RAID 1 has a good read performance but the response time to a write without the fast write cache option enabled is longer than to a non-RAID disk. A RAID 1 array can survive multiple disk failure without loosing data.

The FC 6225 Advanced SerialRAID adapter card supports twin-tailed RAID 1 configuration without fast write cache and single host configuration with fast write cache. The new FC 6230 Advanced SerialRAID adapter can be attached to two hosts with fast write cache. You can configure an even number of disks—up to 72 for a single RAID 1 array. These disks can be configured in two separate domains. A domain contains one copy of the mirrored data. The adapter card ensures that the systems cannot operate independently on different domains when the SSA loop is cut off between the two domains.

An SSA RAID 1 array can automatically replace a failing disk to a hot spare disk. Each hot spare disk can be assigned to a specific domain.

10.1.1.6 RAID 5 in SSA

RAID 5 is a good compromise between high availability and performance. It distributes data sectors among the disk and stores each data block's redundant information on different disks. The RAID 5 array is supported on PCI SSA Multi-Inititator/RAID EL adapter (FC 6215), Advanced SerialRAID adapter (FC 6225 and 6230), and Micro Channel SSA Multi-Inititator/RAID EL adapter (FC 6219). The number of adapters in the same loop is limited to two. An SSA RAID 5 array can contain 3 to 16 member disks. It supports the automatic hot spare function. The biggest disadvantage of RAID 5 is its modest disk write throughput. The fast write cache option can considerably increase the speed and the response time of the disk writes.

10.1.1.7 RAID 10 on an Advanced SerialRAID adapter

RAID 10, or RAID 0+1, is a combination of RAID 0 and RAID 1 technology. The data is striped across a number of mirrored disks. It provides outstanding read/write performance. This storage method is a good choice for speed and availability.

RAID 10 is supported on the Advanced SerialRAID adapter (FC 6230 only) in a one or two hosts configuration. An SSA RAID 10 array can contain 4 to 16 disks in two domains. A domain contains one copy of the mirrored data. The adapter card ensures that the systems cannot operate independently on different domains when the SSA loop is cut off between the two domain. An SSA RAID 10 array can automatically replace a failing disk to a hot spare disk. Each hot spare disk can be assigned to a specific domain.

10.1.2 Comparison of AIX LVM mirror and SSA RAIDs

Table 8 and Table 9 on page 154 summarize and compare the attributes of the different kinds of storage methods.

Table 8 shows a comparison of the AIX LVM features.

Table 8. Comparison of AIX LVM features

	Striping	**Mirroring**	**Striping and mirroring**
Description	Data distributed across disks	All data copied two or three separate disks	Data is stripped across a number of mirrored disks
Disk drives	N	2N or 3N	2N or 3N
Data availability	No redundant information	2 or 3 copies	2 or 3 copies

	Striping	**Mirroring**	**Striping and mirroring**
Suitable for HACMP?	no	yes	yes
Read performance	very high	average	very high
Write performance	very high	modest	high
Hot spare	no	With SSA hot spare tool if all disks are SSA	no

Table 9 shows a comparison of the SSA RAID features.

Table 9. Comparison of the SSA RAID features

	RAID 0	**RAID 1**	**RAID 5**	**RAID 10**
Description	Data distributed across disks	All data copied to two or three separate disks	Distributes data sectors among the disk and stores CRCs	Data is stripped across a number of mirrored disks
Disk drives	N	2N	N+1	2N
Data availability	No redundant information	2 copies	1 copy + CRC	2 copies
Suitable for HACMP?	no	yes	yes	yes
Can be used with optical extender to prevent from site failure?	no	yes	no	yes
Read performance	very high	average	average	very high
Write performance	very high	average	modest	high
Max. number of adapters in one loop	1	2	2	2

	RAID 0	RAID 1	RAID 5	RAID 10
Max. number of disk per RAID array	16	72	16	16
Hot spare	no	hardware	hardware	hardware

10.1.3 SSA hot spare tool

The SSA hot spare tool can be used with AIX LVM mirrored SSA disk pools. It automatically removes the failing disk from the volume group and replaces it with a hot spare disk. The system administrator needs only to replace the bad disk with a new one. The SSA hot spare tool works with all SSA disk units and adapter cards.

This tool has some limitations:

- All SSA disks in the monitored loops have all the logical volumes mirrored and are on separate physical volumes.
- Cannot be used with concurrent volume groups.
- The hot spare tool must have `rsh` access to all nodes connected to the monitored SSA loops.
- Does not support boot or dump devices.
- It can handle only single disk failures in a volume group.

You can obtain the SSA hot spare tool and its installation guide from the following Internet address:

`http://www.hursley.ibm.com/~ssa/rs6k/`

10.2 Site protection with fiber-connected disks

The new Enterprise Storage Server (Shark) and the Advanced SSA Optical Extender make it possible to separate the cluster nodes up to 10 kilometers. With fiber-connected disks, an HACMP cluster is no longer limited to one room. This solution can prevent real-site failure. Though it provides limited protection against extensive disasters, such as earthquakes, it is a cost-effective way of utilizing the extra benefits of the HACMP and SSA disk subsystems.

This disaster recovery solution is based on the fiber-connected disks and virtual LAN. HACMP does not require special configuration to achieve site protection, all the tricks are in the ESS or SSA Optical Extender and the LAN connection between the nodes. The fiber-connected disks and the virtual LAN eliminate the physical distance between the cluster nodes, so the HACMP system, in essence, can act as a single room-based cluster.

The fiber-connected SSA disk acts like a normal SSA disk. To achieve bulletproof site protection, store your mission-critical data on shared and mirrored SSA disks. Each site has to have a full copy of all shared logical volumes. The mirroring is done by either AIX LVM or SSA hardware RAID.

The ESS utilizes the *Peer-to-Peer Remote Copy*. PPRC maintains a synchronous copy (always up-to-date with the primary copy) of data in a remote location. This backup copy of data can be used to quickly recover from a failure in the primary system without losing any transactions.

The HACMP does not support WAN connections between the cluster nodes. For this reason, the cluster nodes on both sites must be connected to a virtual LAN. The virtual LAN is a wide area extension of the traditional local area network. Usually, a virtual LAN consists of high-speed switches connected to an ATM or fiber optic backbone. All the elements of a virtual LAN can use the same logical IP network or subnet. We discuss the network considerations later in this chapter.

An example of the above mentioned HACMP-based site protection solution can be seen on Figure 88.

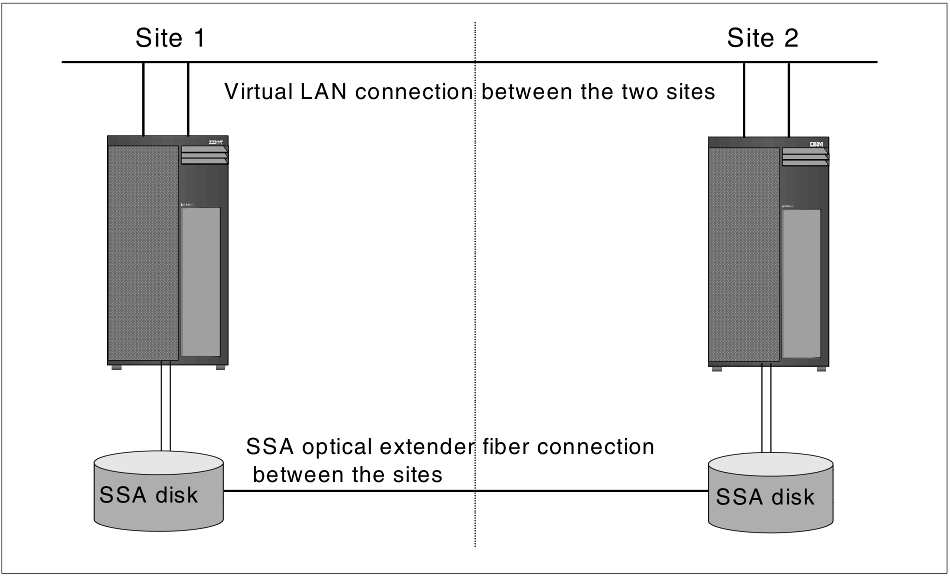

Site 1
Site 2
Virtual LAN connection between the two sites
SSA optical extender fiber connection between the sites
SSA disk
SSA disk

Figure 88. HACMP and fiber-connected SSA disk-based site protection example

In the following sections, we describe in detail the fiber-connected disk and network configuration considerations.

10.2.1 Comparing the HAGEO and HACMP site protection

HACMP with fiber-connected disks is a real alternative to HAGEO because:

- It is a geographic cluster: The distance between the nodes can be up to 10 kilometers (Advanced SSA Optical Extender) or 103 kilometers (ESS).
- Provides hardware-based data protection: The data replication is done through hardware or AIX LVM via the SSA Optical Extender or ESS PPRC.
- Provides disk takeover: HACMP can mount mirrored data disks after a failover even if one side of the mirror is lost.
- Supports IPAT: HACMP can be configured for IP address takeover (IPAT) over long distance virtual LAN.

The main advantage of the HACMP-based site protection as compared to HAGEO is that the data mirroring is done by hardware. Therefore, the data mirroring does not affect the system performance like the software-based Geomirror. But HAGEO provides up to three copies of the site (that is 6-way mirroring!) and data are in different volume groups in each site, while HACMP with SSA disks supports only 2-way mirroring, and the data are in the same volume group across the sites.

When planning a disaster recovery solution, always consider what it protects against and the key risk elements. HACMP is suitable for disaster caused by fire, sabotage, explosion, and other disasters with a relatively small scale. However, HACMP-based site protection does not provide enough protection against large scale disasters, such as earthquakes, floods, or war. Also, in some places, the 10 to 100 kilometers distance is not enough to prevent power failure because both sites may be connected to the same power plant.

Table 10 compares the main features of the HAGEO- and the HACMP-based site protection solutions.

Table 10. Main features of HAGEO- and HACMP-based site protection

	HACMP and fiber connected SSA	**HACMP and mirrored ESS**	**HAGEO**
Maximum distance between the nodes	10 kilometers 32800 feet	103 kilometers (ESCON)	Unlimited

	HACMP and fiber connected SSA	**HACMP and mirrored ESS**	**HAGEO**
Connection for data replicating	Min. 4, typical 8 black fiber	Min. 2 black fiber	High bandwidth, low latency, IP-based WAN
Non-IP based network	TMSSA	Modem based, requires one phone line per site	Modem based, requires one phone line per node
Data mirroring	AIX LVM or SSA RAID 1, 10	ESS Peer-to-Peer Remote Copy	Geomirror
Maximum number of copies	3	2 per site (RAID 1)	3 per site
Performance impact	None	None	15 to 30 percent CPU
Protection against site failure	Yes	Yes	Yes
Protection against extensive disasters	No	No	Yes

10.2.2 Enterprise Storage Server and PPRC

The Enterprise Storage Server (ESS) provides disaster recovery solutions with the Peer-to-Peer Remote Copy (PPRC).

The PPRC feature allows synchronous copying of a physical volume on one storage server to another volume on a second storage server. PPRC ensures write operations on both local and remote copies. You need to connect the ESSNet to both the primary and recovery sites via your LAN in order to initiate PPRC operations.

Note

The PPRC feature requires that you install ESCON adapters on the ESS.

The PPRC feature provides facilities to secure data on a distant server, and the facility is easy to integrate into a existing system since its function is transparent to the system and applications. The key elements are:

- The primary storage server (application site) establishes the paths and links to a secondary storage server (recovery site). The primary storage server also manages the synchronous copying of the volume from the primary site to the secondary site. The remote secondary server can be up to 103 kilometers from the primary server.

- From the primary host, the I/O operation to the secondary storage server is transparent.

Although the storage server manages virtually all the operation, Copy Services provides the means to:

- Initiate and terminate the functions
- Suspend a copy operation
- Query the status of volumes

Also, with PPRC, a recovery host has the ability to recover control of the secondary volume. The PPRC can be set up from StorWatch.

When planning to install ESS in an HACMP cluster, consider the following:

- ESS currently supports SCSI, Gigabit fiber channel, and SAN Data Gateway attachment. In an HACMP configuration, all systems must connected to an ESS either fiber via the SAN gateway or exclusively through SCSI; mixed configurations are not supported.
- Each fiber channel server connection requires its own unique SAN Data Gateway. The SAN Data Gateway cannot be used for any other attachment.
- ESS supports only two-way HACMP configuration.
- ESS and PPRC can store the data in RAID 1 and RAID 5. We suggest you use RAID 5. ESS has at least a 6 GB read/write cache, so RAID 5 is very fast—its write speed is comparable with the speed of RAID 1.
- ESS can automatically avoid disk errors by its internal sparing function.

You can read more about ESS in the *IBM Enterprise Storage Server Introduction and Planning Guide*, GC26-7294.

Figure 89 on page 160 shows a disaster recovery solution based on an HACMP- and ESCON-connected ESS disk subsystem.

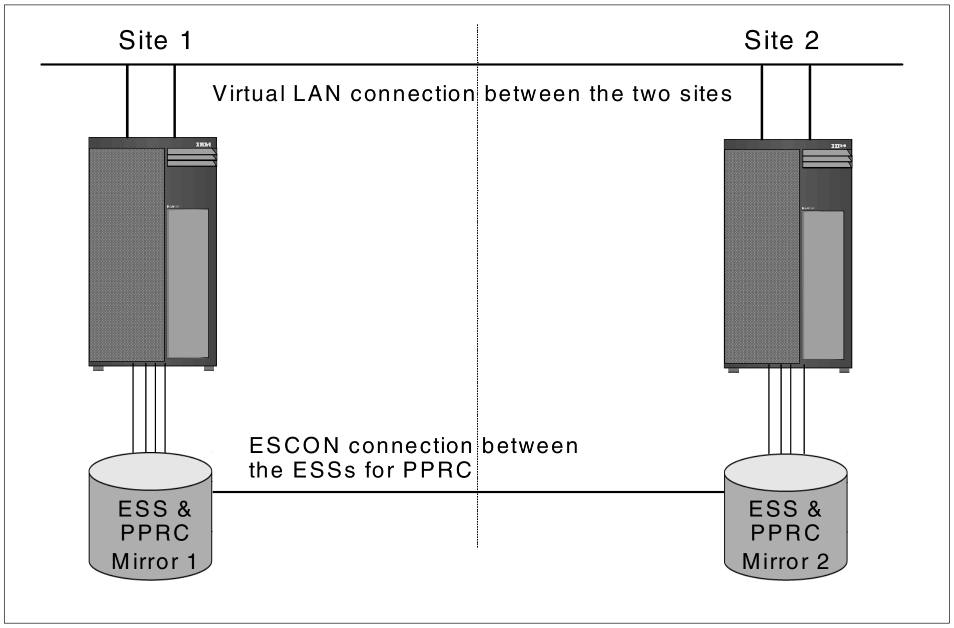

Site 1
Site 2
Virtual LAN connection between the two sites
ESCON connection between the ESSs for PPRC
ESS & PPRC Mirror 1
ESS & PPRC Mirror 2

Figure 89. Disaster recovery with ESS

10.2.3 Advanced SSA Optical Extender

The Advanced SSA Optical Extender connects SSA devices through pairs of fiber optic cables. The Advanced SSA Optical Extender can be plugged into the SSA connector of an SSA disk drawer or host adapter. The optical extender supports both single-mode and multimode operations. Table 11 shows the maximum fiber optical cable lengths between the SSA devices.

Table 11. Advanced SSA Optical Extender fiber optic cable lengths

Operation mode	**Connected devices**	**Max. fiber optic cable length**
Single-mode	7133 Model D40/T40, Advanced SerialRAID adapter (FC 6225, 6230)	10 kilometers 32800 feet
Single-mode	Any type of SSA drawers and adapters	2.4 kilometers 7874 feet
Multimode	7133 Model D40/T40, Advanced SerialRAID adapter (FC 6225, 6230)	3 kilometers 9842 feet

Operation mode	Connected devices	Max. fiber optic cable length
Multi-mode	Any type of SSA drawers and adapters	2.4 kilometers 7874 feet

The Advanced SSA Optical Extender supports all kinds of AIX LVM and SSA hardware RAID data protection technology. If you would like to implement real-site failure protection, your options are:

- AIX LVM mirroring
- SSA hardware RAID 1
- SSA hardware RAID 10

The SSA hardware RAID mirroring can be used only in two-node configurations, and it allows only one adapter per node.

Important

To protect your data from site failure, always ensure that each copy of the mirrored data or the disk mirror domains are on separate sites. Also, you must ensure that you have quorum in the disk volume group, that is, having more than 50 percent of the disks available. It is not possible to have more than 50 percent of the disk in a configuration with fiber extenders and the disks placed on two locations. Therefore, the quorum must be turned off (to avoid the vg from varying off at site 1 if the SSA drawer becomes unavailable at Site 2).

The advanced SSA Optical Extender can be used for cluster TMSSA heart-beat network just like a normal SSA network.

The fiber-connected SSA configuration shown in Figure 90 on page 162 can protect data against site failure. AIX LVM 2-way mirroring is configured for data protection. The first mirror is located on site 1, the second one is on site 2. The drawers are attached by Advanced SSA Optical Extenders and fiber optic cables. The adapter card logic ensures that the two systems can operate independently on different drawers when the SSA loop is cut off between the two drawers. If a host powered down or one of the sites fails, the SSA drawer will close the relevant bypass loop. In Figure 90 on page 162, we have two hosts with two adapters each. The two systems are connected to two SSA drawers placed on different locations and connected with SSA optical extenders. Each SSA drawer has one loop that connects the drawer with all four adapters. If the loop is broken or one or more adapters fails due to a site error or other error, the loop will be set to bypass mode as loop four and five at both drawers.

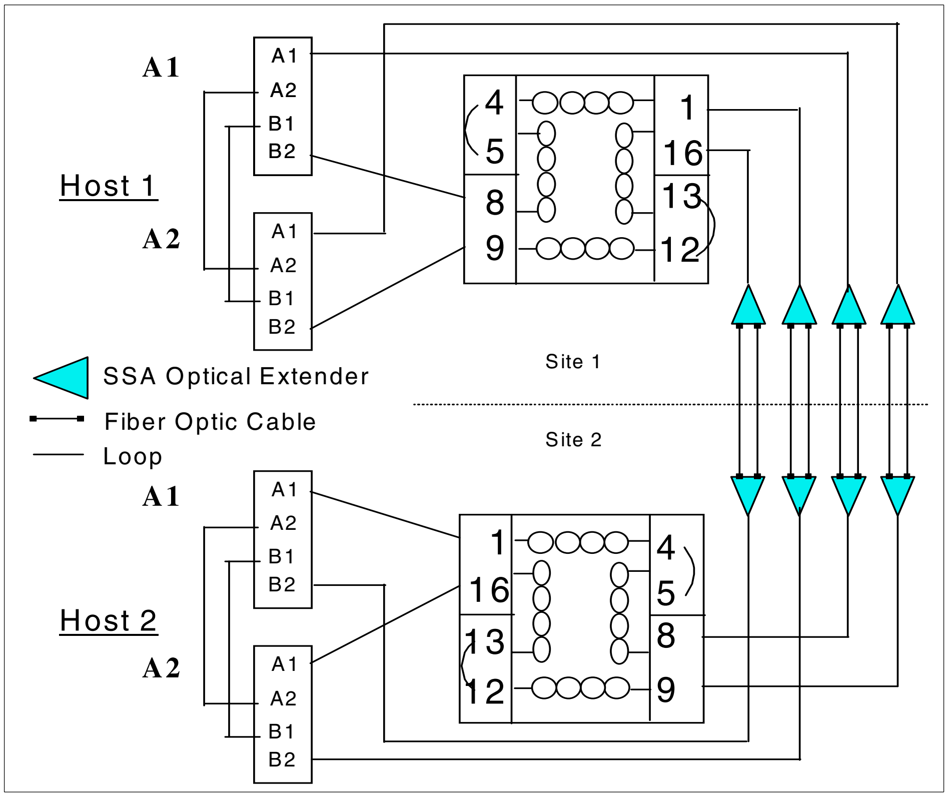


Figure 90. Fiber-connected SSA configuration

10.2.4 Network considerations

This cluster configuration needs special network considerations, because the relatively long distance between the HACMP nodes.

First of all, let's see the cluster heart-beat network. Neither the RS-232 serial line nor the target mode SCSI network can be used for such a long distance. The only solution is the Target Mode SSA network.

The site-to-site IP networks are commonly WAN-based and they include routers. If possible, try to avoid using routers in the network between the cluster nodes. Although it is not officially supported, you can use router-based connections inside an HACMP cluster. In this case, setup your routers as bridges or enable proxy ARP for the appropriate subnets.

We suggest that you implement a virtual LAN between the two sites. The virtual LAN or VLAN extends the local area network by linking the active network devices to a backbone. This backbone is usually a fast-speed, low-level protocol network, such as ATM or STM. The backbone can connect dozens of LANs from separate geographical areas into one big network without using routers. The VLAN-attached devices can be on the same logical IP network (subnet) and they can use traditional LAN-based applications, such as IPX/SPX and NetBios.

On each site, connect the cluster nodes to a switch that is capable of ATM or fiber optic backbone connection. The switch can be any kind, such as ATM, 100 Mb Ethernet or token ring. It is important to connect the switches directly to the backbone without any router, because it enables them to use the same logical IP network (subnet) on both sites. This solution requires one or two more fiber optical cables between the sites but it really simplifies the cluster network configuration. Also, the network switch can provide the required bandwidth to the servers. Finally, connect your existing local client networks, WAN links, and routers directly to the virtual LAN on each site. The remote locations can be linked to both sites. For example, a remote location may have a primary link to site 1 and a lower bandwidth backup connection to site 2.

Important

All HACMP cluster nodes must be on the same logical IP network (subnet).

Figure 91 on page 164 shows a highly available network configuration.

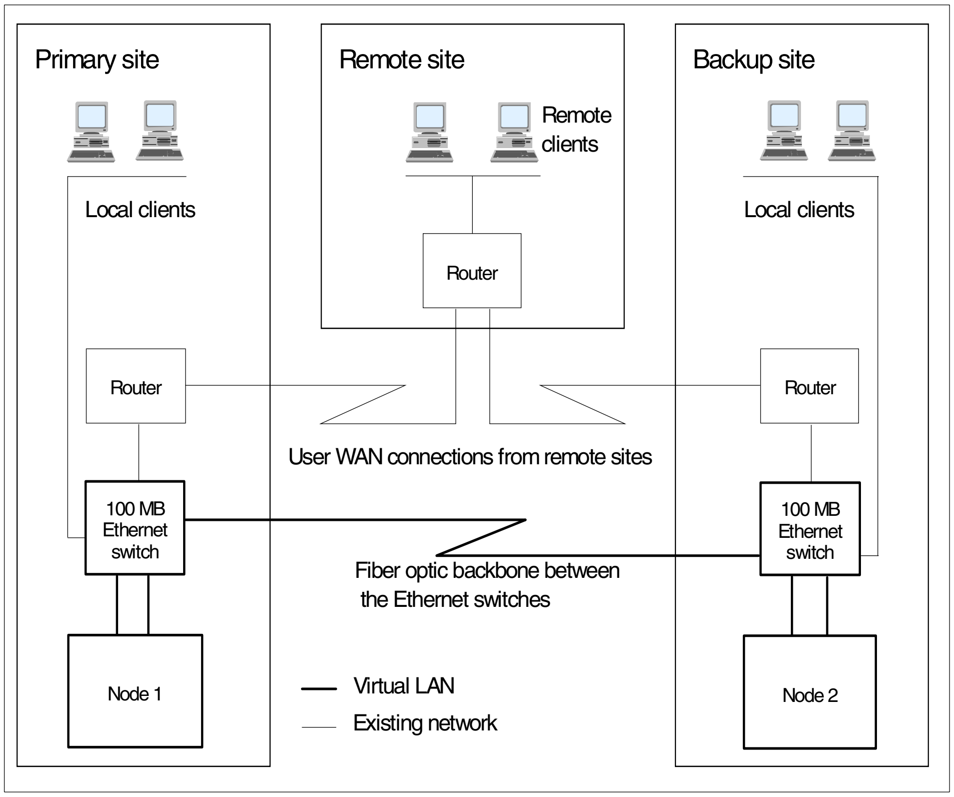


Figure 91. Highly available network configuration

This configuration provides a high availability network for the cluster resources. The cluster nodes, the switches, the backbone, and the local clients are on the same virtual LAN—they use the same IP subnet. If the primary node fails, the primary site's users can connect to the backup node through the backbone. If the entire primary site fails, the remote users can log in to the surviving site via their backup link.

Chapter 11. HACMP and TCP/IP

A well-designed network is the key component of an HACMP cluster. Although implementing a real highly available network is almost impossible, the network environment must be as highly available as possible. The network also must be capable of handling IP address takeover and hardware address takeover.

Hardware address takeover eliminates the need to flush the ARP cache of the network devices. It makes it easier for clients to reconnect to the cluster after a takeover.

Network swapping, discussed in the second part of this chapter, provides a failover method for private networks.

11.1 HACMP-supported network types

Table 12 shows the HACMP-supported network types and their main attributes.

Table 12. HACMP-supported network types

Name	HACMP Network Type	HACMP Network Attribute	HW Address Takeover Support
10 Mb Ethernet	Ether	Public	Yes
100 Mb Ethernet	Ether	Public	Yes
Gigabit Ethernet	Ether	Public	Yes
Token Ring	Token	Public	Yes
FDDI Single Ring	Fddi	Public	Yes
FDDI Dual Ring	Fddi	Public	Yes
ATM Classic IP	Atm	Private	Yes - if the adapters are connected to the same switch
ATM Ethernet Lan Emulation	Ether	Public	Yes - if the adapters are connected to the same switch
ATM Token-Ring Lan Emulation	Token	Public	Yes - if the adapters are connected to the same switch

Name	HACMP Network Type	HACMP Network Attribute	HW Address Takeover Support
SP Switch	Hps	Private	No
Serial Optical Channel Converter	Socc	Private	No
SLIP	Slip	Private	No

The Serial Optical Channel Converter (SOCC) and SLIP are not supported in HACMP/ES.

11.2 Hardware address takeover

Hardware address swapping works together with IP address takeover. The hardware address or MAC address is a unique factory-defined address that ensures that no two adapters have the same network address. The hardware address swapping maintains the binding between an IP address and a hardware address. This eliminates the need to flush the ARP cache of clients and active network devices after an IP address takeover. The hardware address takeover is supported only on Ethernet, token-ring, FDDI, and ATM adapters. The hardware address takeover takes about 60 to 120 seconds.

The hardware address takeover is done by an alternative hardware address. You must specify an alternative hardware address for each HACMP service interface. The alternate address is assigned for the service interface but not for the adapter card. Every time HACMP starts or a takeover occurs, the active service interface gets the alternative hardware address.

11.2.1 Selecting an alternate hardware address

Basically, you can define any kind of alternate address similar in form to the one the manufacturer assigned to the adapter, but we suggest following the recommendations for the adapter cards.

Use the `netstat -i` command to determine an adapter's manufacturer default hardware address. An example output of the `netstat -i` command can be seen in Figure 92 on page 167.

```
dodge:root:/>netstat -i
Name  Mtu   Network     Address            Ipkts Ierrs    Opkts Oerrs  Coll
lo0   16896 link#1                          81458     0    81466     0     0
lo0   16896 127         loopback            81458     0    81466     0     0
lo0   16896 ::1                             81458     0    81466     0     0
tr0   1492  link#2      10.0.5a.c9.1.3b    235541     0   147020     0     0
tr0   1492  9.3.187     dodge              235541     0   147020     0     0
tr1   1492  link#3      10.0.5a.a8.d1.f3   164485     0    90011     0     0
tr1   1492  10.1.1      dodge_sby          164485     0    90011     0     0
en0   1500  link#4      2.60.8c.2f.47.42     2275     0     1619     0     0
en0   1500  192.168.1   dodge_en             2275     0     1619     0     0
at0   9180  link#5      8.0.5a.99.10.4d       130     0      211     0     0
at0   9180  192.168.2   dodge_atm             130     0      211     0     0
```

Figure 92. Sample output of the netstat -i command

In this example, the tr0 adapter’s hardware address is 10.0.5a.c9.1.3b.

11.2.1.1 Specifying an alternate Ethernet hardware address

To specify an alternate hardware address for an Ethernet interface, substitute the MAC address last pair of characters. For example:

Original address:	2.60.8c.2f.47.42
New address:	2.60.8c.2f.47.01

11.2.1.2 Specifying an alternate token-ring hardware address

To specify an alternate token-ring hardware address, set the first two digits of the address to 42. A token-ring hardware address beginning with 42 indicates that the address is set locally. For example:

Original address:	10.0.5a.c9.1.3b
New address:	42.0.5a.c9.1.3b

11.2.1.3 Specifying an alternate FDDI hardware address

To specify an alternate hardware address for a FDDI interface, use 4, 5, 6, or 7 as the first digit of the new address and leave the remaining digits as the default value. You can use any number for the second through sixth digits. For example:

Original address:	10.00.5a.b8.89.57
New address:	40.00.5a.b8.89.57

11.2.1.4 Specifying an alternate hardware address for ATM

The following procedure applies only for the ATM Classic IP interface. The hardware address swapping for ATM LAN Emulation adapters works just like hardware address swapping for the emulated Ethernet or token-ring card.

Therefore, if you use Lan Emulation, follow the appropriate adapter description.

The ATM adapter hardware address is 20 bytes in length. The first 13 bytes are assigned by the ATM switch. The next 6 bytes identify the adapter card. The `netstat -i` command shows only this 6 bytes. The last digits represent the interface number. The alternate ATM adapter hardware address is a total of 7 bytes and it consists of the last 7 bytes of the full ATM adapter hardware address.

To specify an alternate hardware address for an ATM Classic IP adapter, specify a value in the range of 40.00.00.00.00.00 to 7f.ff.ff.ff.ff.ff for the first 6 bytes and use the interface number as the last byte.

For example, `netstat -i` shows that the hardware address of the at1 interface is 08.00.5a.99.08.91, then you can specify an alternate hardware address, such as 50.00.00.00.00.01.

Note

Hardware address swapping for ATM (Classic IP or LAN Emulation) requires that all adapters that can be taken over for a given service address be attached to the same ATM switch. In this case, the ATM switch becomes a single point of failure.

Requirements for ATM hardware address swapping:

- The number of ATM service adapters supporting hardware address swapping can be no more than seven.
- The number of ATM standby adapters configured for hardware address swapping can be no more than seven.
- Each service interface must have an unique ATM interface number.
- You must configure the appropriate ATM interfaces on the standby adapters.

11.2.2 Avoiding network conflicts

When defining a hardware address for a network adapter, ensure that the defined address does not conflict with the hardware address of another network adapter in the network. You can check this with the `ping` command. If you receive duplicated packets from the node configured for hardware address swapping, then you specified an address already in use. In this case, stop the cluster, change the hardware address, and restart the cluster.

The screenshot illustrated in Figure 93 shows the output of the `ping` command with duplicated packages.

```
bmw:root:/>ping dodge
PING dodge.itsc.austin.ibm.com: (9.3.187.185): 56 data bytes
64 bytes from 9.3.187.185: icmp_seq=0 ttl=255 time=2 ms (DUP!)
64 bytes from 9.3.187.185: icmp_seq=0 ttl=255 time=2 ms (DUP!)
64 bytes from 9.3.187.185: icmp_seq=0 ttl=255 time=2 ms (DUP!)
^C
----dodge.itsc.austin.ibm.com PING Statistics----
1 packets transmitted, 1 packets received, +2 duplicates, 0% packet loss
round-trip min/avg/max = 2/2/2 ms
```

Figure 93. Duplicated ICMP packages

11.2.3 Hardware address swapping workarounds

If you cannot configure hardware address swapping in your cluster, try to implement the following workarounds to help the cluster clients reconnect after a takeover has occurred.

First of all, try to decrease the ARP cache timeout of the active network devices. Usually, the ARP cache time out is set to 2 to 20 minutes; AIXs default timeout is 20 minutes. On a switched network, it is efficient to change the ARP table timeout to a smaller value (for example, 90 seconds) for the switch that connects the cluster nodes. If it does not help, you can change the clients' ARP cache timeout. On AIX, you can do that with the `no` command's arpt_killc parameter. The `/usr/sbin/no -o arpt_killc=2` command changes the AIX ARP cache timeout to two minutes.

Another way to force the removal of the old hardware address from a client ARP table is to ping the client. HACMP provides an automated mechanism for this. Add the host name or IP address of a client host you want to notify to the PING_CLIENT_LIST variable in the `/usr/es/sbin/cluster/etc/clinfo.rc` file. Now, whenever a cluster events occur, clinfo.rc sends an ICMP ECHO request to each host specified in PING_CLIENT_LIST. We suggest you add your main routers' addresses to this list.

With AIX 4.3.3, there are important enhancements and changes to ARP. The implementation now supports RFC2225, where the previous implementation supported RFC1577. The enhancement means that IP clients can switch to a backup ARP sever if the current ARP server fails. The IP clients detect and switch back to the primary ARP server when it again becomes available. AIX 4.3.3 also has new features added to the IP configuration, which is very important for HACMP. Whenever an adapter is configured with a new IP address, the system will send a request to other systems on the local LAN to update the ARP table with the new IP address. This feature will also cause duplicate IP address detection; duplicate IP addresses will be logged in the AIX system log. This new feature in AIX 4.3.3. is called Gratuitous ARP.

11.2.4 Hardware address takeover implementation example

In this section, we guide you through the implementation steps of hardware address swapping. We use the sample cluster configurations shown in Figure 94 on page 170.

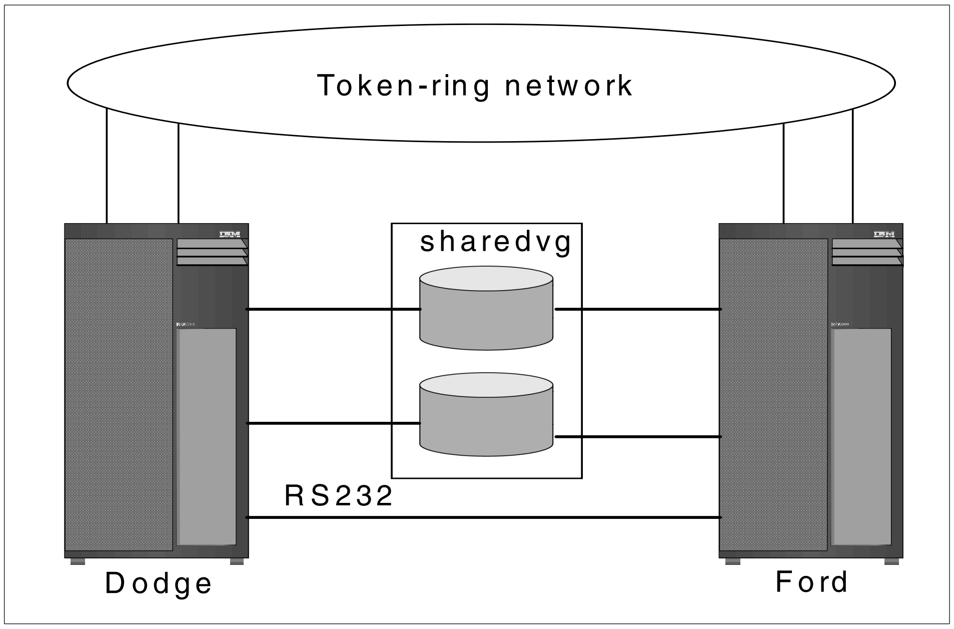


Figure 94. Sample cluster network configuration

An HACMP cluster is set up in mutual takeover mode between two nodes, ford and dodge. The nodes are connected to a token-ring network. The cluster is configured for IP address takeover with interface labels as shown in Table 13.

Table 13. Network interfaces of the sample cluster nodes

Interface	**Node ford**	**Node dodge**
Boot	ford_boot	dodge_boot
Service	ford	dodge
Standby	ford_sb	dodge_sb

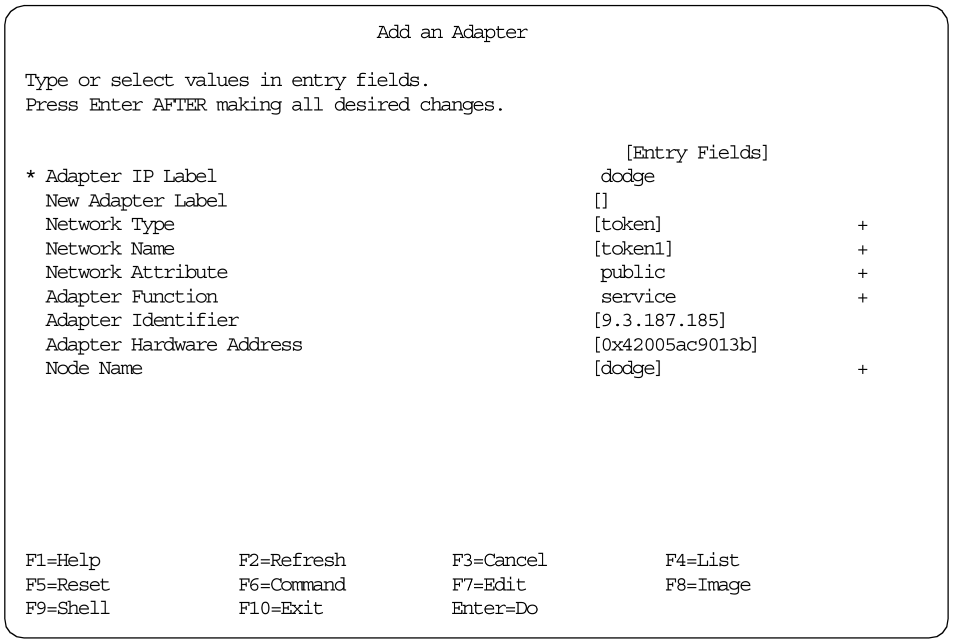
The factory default hardware addresses for the ford and dodge service adapter are as follows. We use the `netstat -i` command to get the hardware addresses.

Dodge:	10.0.5a.c9.01.3b
Ford:	10.0.5a.a8.3c.35

Because ford and dodge have token-ring interfaces, we specify the following alternate hardware addresses:

Dodge:	42.0.5a.c9.01.3b
Ford:	42.0.5a.a8.3c.35

We configure the alternate hardware addresses for HACMP in smit hacmp by selecting **Cluster Configuration -> Cluster Topology -> Configure Adapters -> Add an Adapter** panel. The panel is shown in Figure 95.



```
                                  Add an Adapter

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
* Adapter IP Label                                     dodge
  New Adapter Label                                   []
  Network Type                                        [token]             +
  Network Name                                        [token1]            +
  Network Attribute                                    public             +
  Adapter Function                                     service            +
  Adapter Identifier                                  [9.3.187.185]
  Adapter Hardware Address                            [0x42005ac9013b]
  Node Name                                           [dodge]             +







F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 95. Add an Adapter SMIT panel

You must start the alternate hardware address with `0x`. This indicates that it is a hexadecimal number.

After you successfully configure all the interfaces and the other HACMP parameters, you can start the cluster. Now, you can check the configuration with the `netstat -i` command. A sample output of the `netstat -i` command can be seen in Figure 96 on page 172.

```
dodge:root:/>netstat -i
Name  Mtu   Network     Address            Ipkts Ierrs    Opkts Oerrs  Coll
lo0   16896 link#1                          81458     0    81466     0     0
lo0   16896 127         loopback            81458     0    81466     0     0
lo0   16896 ::1                             81458     0    81466     0     0
tr0   1492  link#2      42.0.5a.c9.1.3b    235541     0   147020     0     0
tr0   1492  9.3.187     dodge              235541     0   147020     0     0
tr1   1492  link#3      10.0.5a.a8.d1.f3   164485     0    90011     0     0
tr1   1492  10.1.1      dodge_sby          164485     0    90011     0     0
```

Figure 96. The modified hardware address of the tr0 interface

11.3 Network swapping

The private networks are used for node-to-node application communication. There is no outside user traffic on them. Usually, the private networks have only one adapter per node, so HACMP cannot protect this interfaces with adapter swapping or IP address takeover. However, HACMP can indicate the failure of a private network: it starts a network down event with the appropriate parameters. We can customize the network down event to protect our private networks with network swapping.

The main idea behind network swapping is that we can reroute the traffic of a failed network to another network. The network swapping adds a new host routing entry for each node's routing table. This new route redirects the failed network's packets to another network.

Network swapping is transparent for the network applications. TCP/IP does not close the connection if network swapping occurs. The network swapping looks like a short duration network outage for a network application. For example, an NFS-mounted file system remains reachable after a network swapping of its network without need to remount it.

To use network swapping, you must enable the ipforwarding with the `no -o ipforwarding=1` command. It is a run-time attribute, so we suggest you place this command in the /etc/rc.net script.

We show how the network swapping works in the following example. For the configuration shown Figure 97 on page 173, we have a token-ring and an Ethernet network between two nodes. The first part of the figure shows the normal way: a packet sent from the node1 token-ring interface goes through the token-ring network and reaches node2 tr0 interface. A packet from node1's en0 to node2's en0 also goes through the Ethernet network.

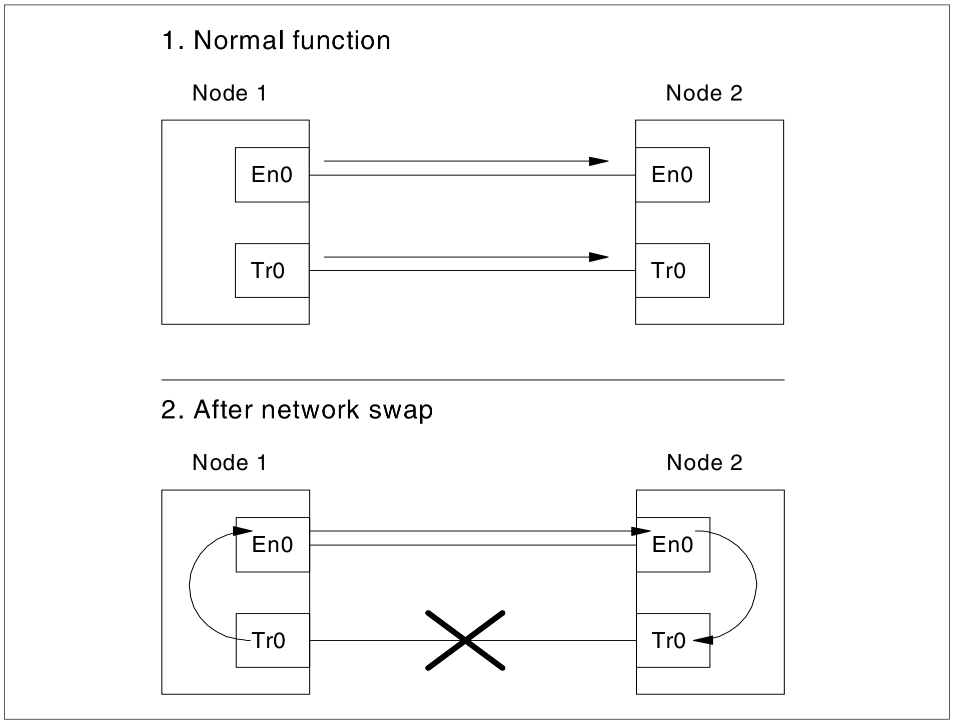


Figure 97. Network swapping example

This sample configuration has the network interfaces shown in Table 14.

Table 14. Sample network interfaces

Network (interface)	**Node1**	**Node2**
Ethernet (en0)	9.138.32.21	9.138.32.22
Token-ring (tr0)	192.168.1.21	192.168.1.22

The default gateway is 9.138.32.254. The netmask is 255.255.255.0 on both networks.

During normal operations, node1's routing table is similar to the following:

```
Destination       Gateway           Flags    If
default           9.138.32.254      UGc      en0
9.138.32/24       9.138.32.21       U        en0
127/8             127.0.0.1         U        lo0
```

```
192.168.1/24     192.168.1.21      U        tr0
```

Node2’s routing table:

```
Destination      Gateway           Flags    If
default          9.138.32.254      UGc      en0
9.138.32/24      9.138.32.22       U        en0
127/8            127.0.0.1         U        lo0
192.168.1/24     192.168.1.22      U        tr0
```

The second part of Figure 97 on page 173 shows the network configuration after the failure of the token-ring network. The network swapping adds a route on node1. Its destination is node2’s token-ring adapter and its gateway is node2’s Ethernet interface. On node2, the new cross route destination is node1’s tr0 card and its gateway is node1’s en0 interface.

Now, the packets from node1’s en0 interface go through the Ethernet network just as in a normal case. The token-ring network is now redirected to the Ethernet, so a packet from node1’s token-ring interface goes to node1’s Ethernet interface, then it goes through the Ethernet network and node2 receives it on its en0 adapter.

Here, you can see the new routing table of node1 after its token-ring network is redirected to the Ethernet interface:

```
Destination      Gateway           Flags    If
default          9.138.32.254      UGc      en0
9.138.32/24      9.138.32.21       U        en0
127/8            127.0.0.1         U        lo0
192.168.1/24     192.168.1.21      U        tr0
192.168.1.22     9.138.32.22       UGH      en0
```

Node2’s routing table:

```
Destination      Gateway           Flags    If
default          9.138.32.254      UGc      en0
9.138.32/24      9.138.32.22       U        en0
127/8            127.0.0.1         U        lo0
192.168.1/24     192.168.1.22      U        tr0
192.168.1.21     9.138.32.21       UGH      en0
```

We highlighted the routing table entries that are added by the network swapping.

If the failed network becomes available again, you can swap back its network traffic by simply removing the cross routes from the nodes’ routing table.

11.3.1 An implementation example using network swapping

This section describes an implementation example of network swapping. Figure 98 on page 175 shows the sample cluster configuration.

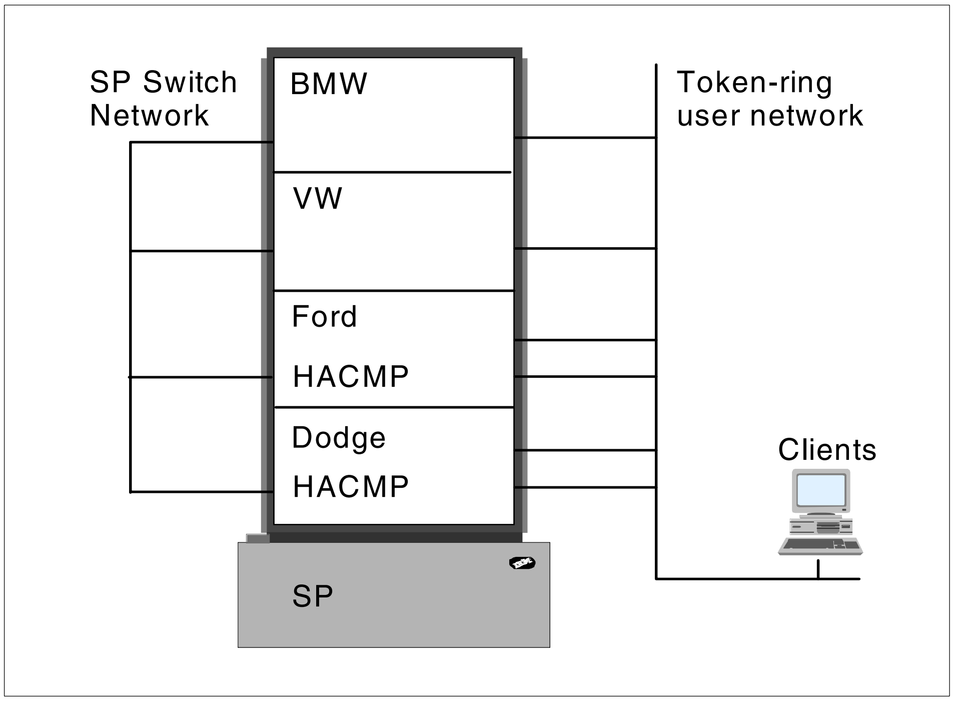


Figure 98. SP cluster network topology

This configuration consists of an SP computer with four nodes. Each node is connected to a token-ring user network as well the SP Switch. We do not care about the SP internal Ethernet (not shown in Figure 98). The node names are dodge, ford, vw, and bmw. Ford and dodge are in an HACMP cluster and have a shared SSA disk pool (not shown in the figure). The TMSSA network is used as a serial network between the two cluster nodes.

Dodge runs the SAP database server; ford is a hot standby node for dodge. Ford and dodge also run application servers. Bmw and vw are SAP application servers. The SAP application servers communicate with the database server through the SP Switch and they also NFS mount two file systems from the cluster’s primary node. The users connect from the token-ring network to any of the application servers. If one application server node fails, the users can reconnect to the other nodes without noticing the error using the SAP internal load leveler function. Therefore, the application servers don't need to be included in the HACMP cluster.

In the shared volume group, we have a few file systems, two of which are NFS mounted on the backup node as well as on the two other nodes. The NFS uses the SP Switch network.

The SP Switch is a key component of this cluster. If it fails, the application servers cannot connect to the database server, and in the end, the application stops. We customize the network down event to achieve network swapping from the SP Switch to the users' token-ring in case of a total SP Switch failure. Because the application server nodes (vw and bmw) depend on the SP Switch, we also implement network swapping for their SP Switch interfaces.

Table 15 contains the sample HACMP network topology for dodge and ford.

Table 15. HACMP network topology

Host name	Network name	Attribute	Function	IP address
dodge	token_ring	public	service	9.138.32.1
dodge_boot	token_ring	public	boot	9.138.32.11
dodge_stby	token_ring	public	standby	192.168.32.1
dodge_sw	sp_switch	private	service	192.168.1.1
dodge_sw_boot	sp_switch	private	boot	192.168.1.11
ford	token_ring	public	service	9.138.32.2
ford_boot	token_ring	public	boot	9.138.32.12
ford_stby	token_ring	public	standby	192.168.32.2
ford_sw	sp_switch	private	service	192.168.1.2
ford_sw_boot	sp_switch	private	boot	192.168.1.12

The network mask on the token-ring network is 255.255.252.0. The SP Switch's netmask is 255.255.255.0.

Table 16 shows the interfaces of bmw and vw (they are not included in the HACMP topology).

Table 16. The network interfaces of vw and bmw

Host name	Network	IP address
vw	Token-ring	9.138.32.3
vw_sw	SP Switch	192.168.1.23
bmw	Token-ring	9.138.32.4
bmw_sw	SP Switch	192.168.1.24

11.3.2 Configuring the SP Switch adapter in HACMP

The SP Switch boot and service addresses will be alias address on the css0 interface. The base IP address, stored in the System Data Repository (SDR) cannot be configured as a service adapter if the SP Switch will be used with IP address takeover. The alias boot and service addresses will appear as ifconfig alias addresses to the css0 IP interface and to the base IP address. The SP Switch's service and boot addresses can be in the base address's subnet. Standby adapter addresses are not used for SP Switch IP address takeover. The address resolution protocol (ARP) must be enabled for the SP Switch.

The base SP Switch addresses of the cluster nodes is shown in Table 17.

Table 17. Base SP Switch addresses

Host name	IP address
dodge_sw_base	192.168.1.21
ford_sw_base	192.168.1.22

11.3.3 Implementing network swapping

We assume that you successfully created the HACMP topology and resources based on the pervious configuration. Therefore, we can customize the cluster now.

Note

We suggest that you do *not* modify the original event scripts. Instead, create pre- and post-event scripts for the HACMP events and store them in a different directory, such as /usr/ha. This prevents you from accidentally overwriting the customized scripts if you install HACMP fixes or upgrades.

HACMP starts the network down event if the SP Switch fails. We do not care about this if it is only an SP Switch adapter failure or a total SP Switch error—in both cases we execute the network swapping. We also need to perform the network swapping on the non-clustered application server nodes, because they connect to the database server node through the SP Switch. On each SP node, we add a new host route to each node’s SP Switch interface. The gateway for this route will be the destination node’s token-ring interface.

Start ipforwarding on all SP nodes by adding the following command to the /etc/rc.net file:

```
/usr/sbin/no -o ipforwarding=1
```

You must also issue this command from the command line to ensure that the IP packet forwarding is enabled.

We create a network down post-event and the appropriate post-event script, called network_down_post. This script checks the network name passed by the HACMP network down event. If the failing network is not the SP Switch, it terminates. Because the HACMP always starts the network down event on both nodes, and we need to avoid creating multiple routing entries on the nodes, we put an additional checking in the network_down_post script to ensure that the new cross routes are not created two times. If the cluster node owns the primary node’s service address, the post event starts a script on each node by rsh. This script, called mkr, sets up the cross routes on the nodes. Otherwise, the network_down_post script terminates. Also, the script sends an e-mail for the system administrator with an error message. The network_down_post script is the same on both HACMP cluster node.

The following is a sample network_down_post script for the cluster nodes:

```
#!/bin/ksh
##########################################################################
# /usr/ha/network_down_post
# If the switch fails this script starts the /usr/ha/mkr script on
# each node.
# This script receives the following parameters from the
# network down event:
# $1: event name
# $2: event error return code
# $3: node name or -1 if it is a global network down
# $4: network name
# The HACMP topology name of the SP Switch network:
NETWORK=sp_switch
# Temporary file to store some stuff to create an E-mail message
```

```
TMPFILE=/tmp/message
# Node1's (dodge) service address
SERVICE=dodge
if [ "$4" = "$NETWORK" ]
then
    # Check, that is it the primary server?
    netstat -i|grep -w $SERVICE>/dev/null 2>&1
    if [ "$?" != "0" ]
    then
        # We are on the backup node, nothing to do from here
        exit 0
    fi
    # send an E-mail to the system administrator
    echo Switch error! >TMPFILE
    echo Please perform problem determination >>TMPFILE
    echo and restart the switch! >>TMPFILE
    mail -s "Switch error!" root <TMPFILE
    # Put some messages into /tmp/hacmp.out
    echo Switch error! Performing network swapping...
    # Start the /usr/ha/mkr script on each node to change the nodes
    # routing table.
    /usr/bin/rsh dodge /usr/ha/mkr
    /usr/bin/rsh ford /usr/ha/mkr
    /usr/bin/rsh vw /usr/ha/mkr
    /usr/bin/rsh bmw /usr/ha/mkr
    echo Network swapping finished! Now you can reach all the Switch
    echo interfaces through the Token-Ring network.
fi
exit 0
```

The mkr script creates the appropriate routing entries for each node's SP Switch interface on every node. These scripts are different on each node, so we have the following four sample scripts for every node.

A sample mkr script for node dodge:

```
#!/bin/ksh
##########################################################################
# /usr/ha/mkr
# This script changes the node's routing table in case of network swapping
# Called from network_down_post script
# Add host route for ford_sw through ford
chdev -l inet0 -a route=host,-hopcount,1,,ford_sw,ford
# Add host route for vw_sw through vw
chdev -l inet0 -a route=host,-hopcount,1,,vw_sw,vw
# Add host route for bmw_sw through bmw
chdev -l inet0 -a route=host,-hopcount,1,,bmw_sw,bmw
```

A sample mkr script for node ford:

```
#!/bin/ksh
##########################################################################
# /usr/ha/mkr
# This script changes the node's routing table in case of network swapping
# Called from network_down_post script
# Add host route for dodge_sw through dodge
chdev -l inet0 -a route=host,-hopcount,1,,dodge_sw,dodge
# Add host route for vw_sw through vw
chdev -l inet0 -a route=host,-hopcount,1,,vw_sw,vw
# Add host route for bmw_sw through bmw
chdev -l inet0 -a route=host,-hopcount,1,,bmw_sw,bmw
```

A sample mkr script for node vw:

```
#!/bin/ksh
##########################################################################
# /usr/ha/mkr
# This script changes the node's routing table in case of network swapping
# Called from network_down_post script
# Add host route for dodge_sw through dodge
chdev -l inet0 -a route=host,-hopcount,1,,dodge_sw,dodge
# Add host route for ford_sw through ford
chdev -l inet0 -a route=host,-hopcount,1,,ford_sw,ford
# Add host route for bmw_sw through bmw
chdev -l inet0 -a route=host,-hopcount,1,,bmw_sw,bmw
```

A sample mkr script for node bmw:

```
#!/bin/ksh
##########################################################################
# /usr/ha/mkr
# This script changes the node's routing table in case of network swapping
# Called from network_down_post script
# Add host route for dodge_sw through dodge
chdev -l inet0 -a route=host,-hopcount,1,,dodge_sw,dodge
# Add host route for ford_sw through ford
chdev -l inet0 -a route=host,-hopcount,1,,ford_sw,ford
# Add host route for vw_sw through vw
chdev -l inet0 -a route=host,-hopcount,1,,vw_sw,vw
```

Now, if an SP Switch error occurs, the HACMP starts the network down event and its post-event script. On the primary node, the post-event script initiates the network swapping by starting the mkr script on each node. This adds the necessary host-specific routes. Finally, the SP Switch interfaces become available again for the TCP/IP application through the token-ring network.

11.3.4 Recovering from SP Switch errors

The network swapping works for one way only. You cannot configure automatic recovery methods for SP Switch errors, because HACMP cannot recognize when the SP Switch is up again. HACMP can only check that an interface is up or down. If the network swap is executed, the css0 interface becomes available through another network, so HACMP can ping it, and in the end, HACMP decides wrongly that the SP Switch is working again and it starts a network up event.

After the SP Switch is up and running, you can manually recover from the network swapping without affecting the cluster. You need to remove only the cross routes from every node.

Here, we provide scripts for fast recovery from network swapping. The recovery procedure works similarly to the network swapping. You must manually start a script on the control workstation if the SP Switch works again. This script is called /usr/ha/switch_up and it starts the/usr/ha/rmr shell script on every node. The rmr script removes the cross routes from each node's routing table. This script is different on each node.

The following is a sample rmr script for node dodge:

```
#!/bin/ksh
##########################################################################
# /usr/ha/rmr
# This script removes the extra routes from the node routing table
# after the switch is working again
# Called from switch_up script from the control workstation
# Remove host route ford_sw through ford
chdev -l inet0 -a delroute=host,,ford_sw,ford
# Remove host route vw_sw through vw
chdev -l inet0 -a delroute=host,,vw_sw,vw
# Remove host route bmw_sw through bmw
chdev -l inet0 -a delroute=host,,bmw_sw,bmw
```

A sample rmr script for node ford:

```
#!/bin/ksh
##########################################################################
# /usr/ha/rmr
# This script removes the extra routes from the node routing table
# after the switch is working again
# Called from switch_up script from the control workstation
# Remove host route dodge_sw through dodge
chdev -l inet0 -a delroute=host,,dodge_sw,dodge
# Remove host route vw_sw through vw
```

```
chdev -l inet0 -a delroute=host,,vw_sw,vw
# Remove host route bmw_sw through bmw
chdev -l inet0 -a delroute=host,,bmw_sw,bmw
```

A sample rmr script for node vw:

```
#!/bin/ksh
###########################################################################
# /usr/ha/rmr
# This script removes the extra routes from the node routing table
# after the switch is working again
# Called from switch_up script from the control workstation
# Remove host route dodge_sw through dodge
chdev -l inet0 -a delroute=host,,dodge_sw,dodge
# Remove host route ford_sw through ford
chdev -l inet0 -a delroute=host,,ford_sw,ford
# Remove host route bmw_sw through bmw
chdev -l inet0 -a delroute=host,,bmw_sw,bmw
```

A sample rmr script for node bmw:

```
#!/bin/ksh
###########################################################################
# /usr/ha/rmr
# This script removes the extra routes from the node routing table
# after the switch is working again
# Called from switch_up script from the control workstation
# Remove host route dodge_sw through dodge
chdev -l inet0 -a delroute=host,,dodge_sw,dodge
# Remove host route ford_sw through ford
chdev -l inet0 -a delroute=host,,ford_sw,ford
# Remove host route vw_sw through vw
chdev -l inet0 -a delroute=host,,vw_sw,vw
```

The following sample shell script (/usr/ha/switch_up) performs the recovery procedure. It starts the rmr scripts on each node. Place this script in the control workstation.

```
#!/bin/ksh
##########################################################################
# /usr/ha/switch_up
# If the switch works again start this script to remove the cross routes
# from the nodes.
# This script calls the /usr/ha/rmr script on every node.
/usr/bin/rsh dodge /usr/ha/rmr
/usr/bin/rsh ford /usr/ha/rmr
/usr/bin/rsh vw /usr/ha/rmr
/usr/bin/rsh bmw /usr/ha/rmr
```

If you start the switch_up, it calls the rmr scripts on each node. The rmr script removes the cross routes from the nodes and the SP Switch interfaces become available through the SP Switch network.

Chapter 12. HACMP and Lotus Domino configuration

When we are planning for high availability Lotus Domino servers, we have a number of technologies to achieve the availability. This section describes the different technologies available for implementing Domino servers in an AIX environment. The technologies covered are:

- Domino cluster
- Domino partitioned servers
- HACMP
- HACMP/ES

Domino clustering and Domino partitioned servers are features of the Domino Enterprise Server license and are available for all platforms including IBM AIX. HACMP and HACMP/ES are IBM products for all AIX platforms, RS/6000, and SP nodes.

12.1 Domino cluster

A Domino cluster is a technology in which two or more Domino servers are connected in a high bandwidth network. The Domino servers maintain copies of databases on the nodes in the cluster. However, each database doesn't need to be on every node in the cluster. As an example, take three servers A, B, and C. The servers have three databases I, II, and III, where I is on server A, II on server B, and III on server C. To improve the availability, the three servers are configured into a Domino cluster *Letters.* The three databases can be distributed with databases I and III on server A, databases I, II, and III on server B, and databases II and III on server C as shown in Figure 99 on page 186. This solution has a high degree of flexibility in that each database can have one, two, or three copies to match the availability requirements for the particular database.

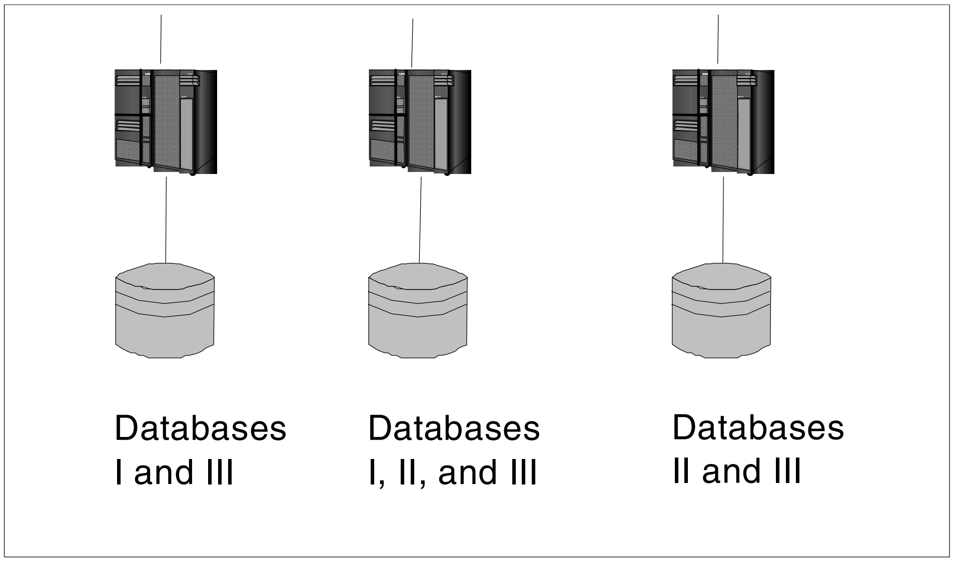


Figure 99. Lotus Domino cluster with three databases

Furthermore, the three servers don't have limitations on the distance between them. As long as you have the required bandwidth in the network, the three servers can be placed anywhere. If a client in the network tries to access a database on a server that is currently unavailable, the request will be redirected to another copy of the database by the transparent failover feature or the load balancing feature of Domino clustering, which can redirect the open request for a database from an unavailable or overloaded server to another available or unloaded server on the cluster. In a clustered Domino environment, replication is event-driven to give a near real-time update, whereas in a standard Domino environment, replication is schedule-driven. For this reason, you must take special considerations into account when planning a clustered environment. Network bandwidth and the total number of replicas per database within the cluster are critical factors that impact the overall performance of the cluster:

- Clustered servers should be interconnected through a high bandwidth network to allow for tight synchronization between the databases in the cluster.
- Frequently accessed read-mostly databases, such as discussion databases, can be spread across several servers in the cluster; but single-owned, read-write databases, such as mail databases, are not likely candidates for Domino clustering.

- Well-designed Domino database documents can reduce the replication traffic between clustered Domino servers. Keep in mind that whenever a field is modified, it has to be pushed to the replicas maintained by the cluster.

For more information about Domino clustering, refer to the Domino online documentation on the product CD-ROM: *Domino Administration Help*, *Extending the Domino System*, and *Setting Up a Cluster*.

12.2 Domino partitioned servers

Domino partitioned servers are a feature of the Domino Enterprise Server license that allows up to six Domino servers to run on a single computer. Partitioning a single computer into separate Domino servers provides the following:

- A reduced number of computers to own and administer in order to support an independent group of users
- Full Domino security between partitioned servers when implementing independent groups of users

Partitioned servers share the same Domino binaries, but each has its own assigned Domino data directory containing the required set of system files, such as Public Address Book, notes.ini, and database files associated with each Domino server. Also, each partitioned server has memory requirements that should be sized according to the nature of the tasks that are going to be running in each of them.

Because not every partitioned server requires its own CPU, it is possible to run partitioned servers both in single CPU systems or in SMP systems with multiple CPUs. High-end configuration systems (with powerful single or multiple CPUs and large amounts of memory) are good candidates for implementing partitioned servers due to Domino's ability to take full advantage of very big and resource-rich configurations. The resource handling will be further enhanced in future versions of the Domino server. For performance improvement results, refer to 4.2, "Performance In Partitioned Domino Servers" on page 101 of the IBM Redbook *High Availability and Scalability with Domino Clustering and Partitioning on AIX*, SG24-5163.

Partitioned servers can share a single network interface adapter (by either using a different IP address for the same adapter or using different IP ports for the same IP address), or use multiple network interface cards (by using a different IP address for each adapter).

Because Domino partitioned servers are running on the same system and under control of the same instance of AIX, the high availability features are limited. Failure in hardware or in AIX will not be handled, only server failures in the Domino server will be managed.

For more information about Domino partitioned servers, refer to the Domino online documentation on the product CD-ROM: *Domino Administration Help*, *Extending the Domino System*, and *Setting Up a Partitioned Server*.

12.3 HACMP/ES and HACMP

High availability cluster multiprocessing/enhanced scalability (HACMP/ES) for AIX Version 4.3.1 and high availability cluster multiprocessing (HACMP) for AIX Version 4.3.1 are control applications that can link up to 32 HACMP/ES or eight HACMP RS/6000 servers or SP nodes into a highly available cluster. HACMP clusters can be configured in several modes for different types of processing requirements:

- Concurrent access mode suits environments where all of the processors must work on the same work load and share the same data at the same time.
- In mutual takeover mode, the processors share the work load and back each other up.
- Idle standby allows one node to back up any of the other nodes in the cluster.

For a Domino server environment, only mutual takeover and idle standby configurations are possible since concurrent access to Domino databases is not supported. For example, in a two-machine scenario with shared disks, one machine can take over the data file systems of the other in case of a failure. By default, HACMP monitoring and takeover scripts only check for machine or network failures. To incorporate the crash of a Domino server in the scripts already provided with HACMP, customized heartbeat programs and takeover scripts must be included. An example of an HACMP installation of a Domino server is shown in Figure 100 on page 189. The two servers, server A and server B, both have the Domino server code installed either on internal disks or on the file system at the shared disk. For further details about how to install and configure Lotus Domino on AIX, see the IBM redbook *Lotus Domino R5 for IBM RS/6000*, SG24-5138.

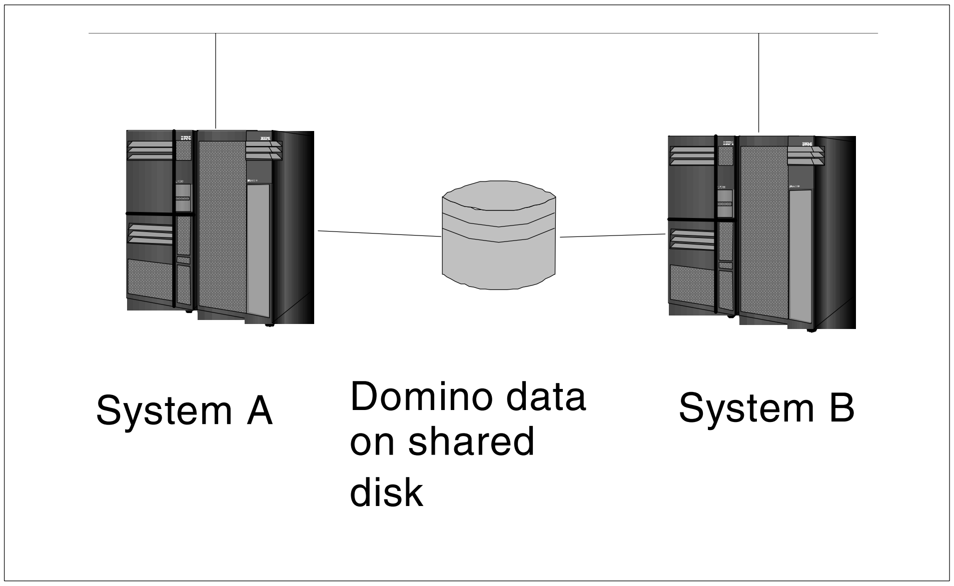


Figure 100. HACMP configuration with Domino data on shared disk

12.4 Domino cluster compared to HACMP/ES and HACMP

From the previous description, it's obvious that there are differences in the configurations depending on the selected cluster technology. Domino offers a SW clustering method that provides great flexibility per database mirroring, and the servers can be placed anywhere without limitation in the distance between nodes. The only requirement is a high bandwidth network connection between the nodes. With Domino clustering, you can also have a mixed environment—not all nodes have to run on the same operating system. One node can run AIX on a RS/6000 and other nodes can run LINUX or NT on a Netfinity server.

An HACMP cluster uses hardware redundancy to achieve high availability: two nodes are able to execute the Domino code but only one is active at any time and the disks are in a RAID configuration, mirrored or RAID 5 to protect against failure. The disk configuration makes the disk I/O more effective, since at any time, there is a direct disk connection to the data disk and you don't have to mirror via a network.

Let's use some examples and discuss which solutions would be best in each situation.

12.4.0.1 A mail server with 500 users

The characteristics of a mail server is that it contains a mail database for each mail user that is, in this case, 500, a high number of databases with a large amount of data. In average mail databases, there are 10 MB for a typical installation. When mail is sent to a user, he or she typically reads it, and for 50 percent of the mail, the receiver would reply with a mail to one or more receivers.

Let's assume that 50 percent of the replies go back to only one person and 50 percent go to two people. If 100 mail messages are sent, 100 are going to be read. This causes 25 sent to one person and 25 sent to two people equaling 50 mails sent and a read-to-write ratio of 57 percent to 43 percent. This is considered a high write ratio. Let's see what happens when 100 mail messages are sent to a server. First, the 100 mails are written into the mail.box server mailbox, then the 100 mails are distributed to the user mail boxes, 100 more writes, and finally the 100 mails are erased from the server's mail box, another 100 writes. For each 100 mail messages we send, 300 are written or erased from databases, or for each mail, we have three database updates.

For each of these operations, the Domino server must trigger an event and synchronize data. The synchronization process involves starting the process, getting the data, accessing the remote server via the network and transmitting the update, receiving acknowledgment for the data, and then clearing the request. All these procedures require CPU time and are resource consuming compared to a disk write operation handled by the disk controller hardware. In a write-intensive environment, such as a mail server with a read-to-write ration of 57 percent to 43 percent , or servers with an even higher write ratio, we recommend using HACMP technology from Domino clustering to achieve high availability.

12.4.0.2 A Domino-based information database (read mostly) server

Typical Domino applications include information servers that serve the data either to Notes clients or to Web browsers. These applications are information servers and help disk-type applications where people ask a question or send a request that is transformed into an information item, which is then read by many. The read-write ratio on this type of application server is typically less than 20 percent write and more than 80 percent read. Domino clustering is an excellent solution for this type of application. The number of writes is low, and the derived performance loss is acceptable because of the advantages achieved by Domino clustering. The advantage we get is scaled performance because the data now resides on two servers and not one—this gives close to double the performance for each server in the cluster, and at the same time, we get high availability since the data can be retrieved from another server should one be unavailable. Furthermore, it is not very complicated to configure a Domino cluster and the servers do not need to be placed very close to each other.

12.4.0.3 Discussion database (read and write) server

A discussion-type application typically has a read-write ratio between a mail server and a read-mostly database that it is more uncertain if one should choose an HACMP solution or base the cluster on Domino technology. It will depend on the actual application whether performance is a issue. HACMP is most likely the best solution since we do not have the Domino synchronizing overhead; otherwise, if performance is not an issue, and the most important concern is to have get a copy of the data, Domino clustering might be a good solution.

12.5 Installation of Domino R5 on AIX 4.3.3 with HACMP

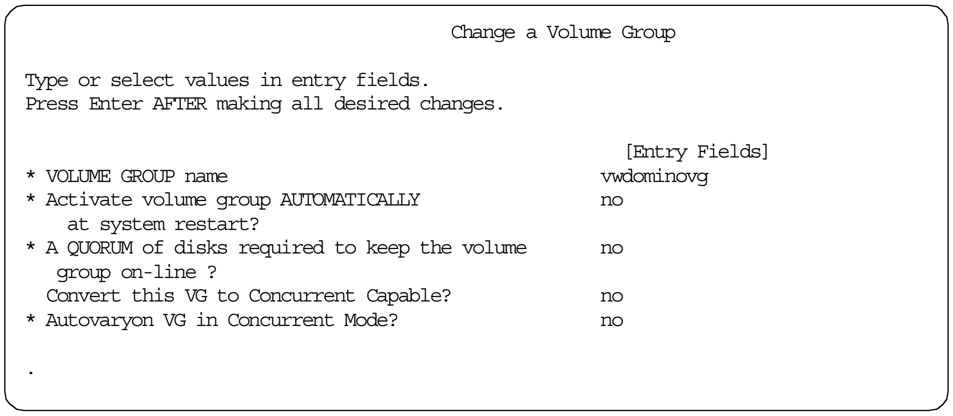
The installation can be done either as an HACMP installation with two nodes both running the Domino server, or with one active node and one takeover node. We assume the first solution is preferred in most cases because the customer can use the second system during normal operation and, therefore, get a higher value for the investment. We assume Domino will be defined as a resource group on both servers, configured in a such way that one server can take over for the other. To configure the resource groups, we must define them as either cascading or rotating resource groups.

In a rotating resource group, all servers in the list of participating servers can acquire the resource group. Initially, when the cluster starts, the first joining node acquires the first available resource group per network, the second node acquires the second, and so on, until all are acquired. When a node fails, the next available node in the list of servers on its boot address acquires the resource group. When the failing server rejoins the cluster, it does not reacquire any resource groups. It rejoins as a standby node.

In cascading resource groups, there is a list of servers called the resource chain, each with a priority for the resource group, just as is the case for a rotating resource group. When a node with higher priority reintegrates into the cluster, it takes back control of the resource group should it be controlled by another node. In this example with Domino severs, we will have two servers normally running on two systems, and in case of system failure on either system, we want the other system to take over. Clearly, to support this function, cascading resource groups are most suited because we want one specific Domino server to run on one specific AIX system in normal operation, and after a system failure, we want the failing server to reintegrate and reacquire the resource group.

The installation of the Domino server code, the definition of Domino domain name, creating users, and so on is straight forward. The procedure is described in the Domino manuals and in the redbook *Lotus Domino R5 for IBM RS/6000*, SG24-5138. The Domino server does not perform a system ID check; therefore, when the Domino installation is completed and tested on one AIX server, it can be taken over and restarted on the other AIX server in the HACMP cluster provided that programs and data are available. The only configuration needed on the second server is copying the /opt link from the server where Domino was installed to the other server. The HACMP cluster installation of Domino is easy because the Domino installation places all the code in one directory tree and all the data in another. Nothing is installed outside these two directories except from the /opt link. The only difference from a standard installation is that we must install the servers as partitioned servers. Partitioned servers are a Domino facility that allows two Domino server instances to run simultaneous on one server, and we will need that in case of takeover. We must also define disk space and resource groups for the servers. For the installation, we use the cluster german on servers bmw and vw. Because both servers in normal mode will be running Domino server simulations, there is no reason to install the binaries on shared disk—the data will be installed on shared disks belonging to two different volume groups in two different resource groups. The two resource groups will be named bmwdominorg and vwdominorg and will, with the volume group contain, an IP label, the service IP address for the system running the Domino server.

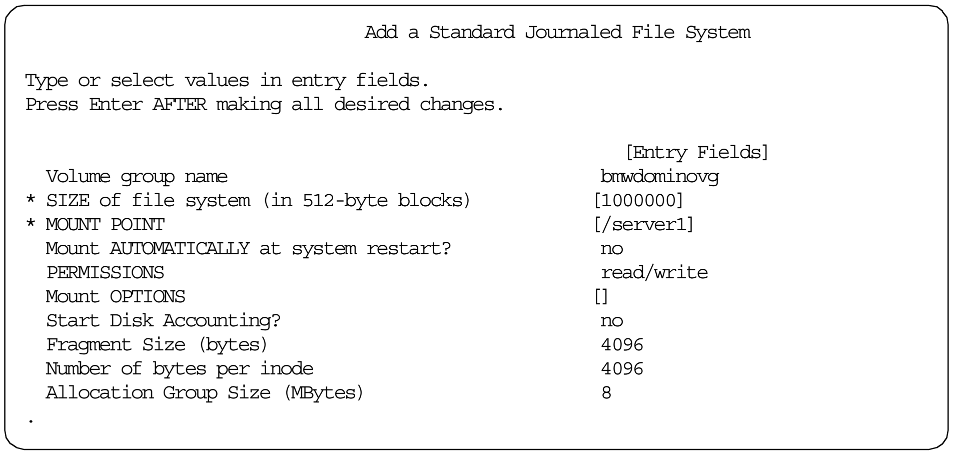
First on server bmw, define the two volume groups bmwdominovg and vwdominovg with smit. Set the volume groups as shown in Figure 101.



```
                                   Change a Volume Group

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                          [Entry Fields]
* VOLUME GROUP name                                     vwdominovg
* Activate volume group AUTOMATICALLY                   no
    at system restart?
* A QUORUM of disks required to keep the volume         no
   group on-line ?
  Convert this VG to Concurrent Capable?                no
* Autovaryon VG in Concurrent Mode?                     no

.
```

Figure 101. Change an AIX volume group

Then, define two file systems, one on each volume group, to hold Domino data. Name the two file systems server1 and server2 to be compliant with the naming standards for the Domino installation on AIX. See Figure 102 on page 193.



```
                           Add a Standard Journaled File System

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                          [Entry Fields]
  Volume group name                                     bmwdominovg
* SIZE of file system (in 512-byte blocks)             [1000000]
* MOUNT POINT                                          [/server1]
  Mount AUTOMATICALLY at system restart?                no
  PERMISSIONS                                           read/write
  Mount OPTIONS                                        []
  Start Disk Accounting?                                no
  Fragment Size (bytes)                                 4096
  Number of bytes per inode                             4096
  Allocation Group Size (MBytes)                        8
.
```

Figure 102. Creating a Domino data directory for server1

For the /server2 file system in vwdominovg, define the file system at bmw, and then later import it on vw. See Figure 103.

```
                                Add a Standard Journaled File System

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                         [Entry Fields]
  Volume group name                                     vwdominovg
* SIZE of file system (in 512-byte blocks)             [1000000]
* MOUNT POINT                                          [/server2]
  Mount AUTOMATICALLY at system restart?                no
  PERMISSIONS                                           read/write
  Mount OPTIONS                                        []
  Start Disk Accounting?                                no
  Fragment Size (bytes)                                 4096
  Number of bytes per inode                             4096
  Allocation Group Size (MBytes)                        8
.
```

Figure 103. Creating a Domino data directory

We must also define a file system for the binaries. According to the standard, the binaries should be installed in /usr/domino. This file system contains binaries and is not very performance demanding. We can, therefore, create the file system in the root volume group on both systems. See Figure 104 on page 194.

```
Add a Standard Journaled File System

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                         [Entry Fields]
  Volume group name                                     rootvg
* SIZE of file system (in 512-byte blocks)             [500000]
* MOUNT POINT                                          [/usr/domino]
  Mount AUTOMATICALLY at system restart?                no
  PERMISSIONS                                           read/write
  Mount OPTIONS                                        []
  Start Disk Accounting?                                no
  Fragment Size (bytes)                                 4096
  Number of bytes per inode                             4096
  Allocation Group Size (MBytes)                        8
```

Figure 104. Creating a Domino program directory

We start the installation on system bmw, and then when it is completed, copy the installation to server vw. However, before we start the installation script, we must complete some pre-installation tasks:

1. Stop the sendmail daemon on both servers:

```
stopsrc -s sendmail
```

and prevent it from starting again:

```
chrctcp -d sendmail
```

2. Stop any http daemons, if running, and disable them in /etc/inittab.
3. Create a group notes on both servers and make sure that the group has the same GID on both servers.
4. Create two user accounts, notes1 and notes2, one for each Domino partition. Make sure that both users have the same UID on both systems.
5. Change the ownership of /server1 and /server2 to be owned by notes1:notes and notes2:notes on both systems.
6. Make sure that AIX allows up to 128 processes per user with the `lsattr -El sys0 | grep maxuproc` command. If needed, change the value with the `chdev - sys0 -a maxuproc=128` command.
7. Install Domino as an Enterprise server and choose the option to let more than one server run on the system. In this example, two servers run on one system. The installation program lets you only type in one AIX user to run both servers. Accept this setting, and then after the first installation phase, change the ownership of all files in /server2 to be owned by notes2 and group notes. Make sure that all files in /server1 are owned by notes1 and group notes.
8. Edit the .profile file in the home directory of notes1 and notes2. Add /server1 to the PATH settings for user notes1, and /server2 for user notes2, and eventually also the binary path /opt/lotus/bin.

Now, after installing Domino, it is time to configure HACMP. Make sure that /usr/domino exists on both servers and that the /server1 and /server2 file systems are defined on both servers owned by notes1 and notes2. Copy all the binaries in /usr/domino from bmw where it was installed to vw.

You can now run the installation program /opt/lotus//bin/http httpsetup and generate IDs for both servers and the Domino administrator.

12.6 Start and stop scripts

To bring the Domino server up and down with the resource group, we need a start and a stop script. These scrips should be as secure as possible, and the recommendation is to keep them simple to minimize the risk of a failure in the script. The start script should ensure that it is executed by the correct user and then start the server. It is important to check for the correct user name, because if Domino is started as another user, it would either not start because the Domino data directory is not defined in the PATH variable, or it starts but then the server may change the ownership of some files in the data directory and the notes user will not be able to start the server again. When the server is started, it can configured to create a log file as the example in Figure 105 on page 196, or the output can be redirected to /dev/null. Normally, there is no reason to keep the log file because the same information is written to the Domino database log.nsf. But in cases where we try to debug the system, it can be useful to have the file as a flat ASCII file.

```
#!/bin/ksh
#
#Shell script to start Domino server in a HACMP environment
#02022000 IBM ITSO Austin
#

#export DEBUG_ENABLE_CORE=1

echo  $LOGNAME | grep notes
if [ $? != 0 ]
then
        echo "not a accepted Domino user ! "
        echo ""
        exit 1
fi

mv /server1/notesserver.log.1 /server1/notesserver.log.1 2>/dev/null
mv /server1/notesserver.log /server1/notesserver.log.1 2>/dev/null
nohup /opt/lotus/bin/server   > /server1/notesserver.log 2>&1 &

exit 0
```

Figure 105. Domino server start script

To stop the server, a -q option is provided for the server command. However, we want to make sure that the server is down and all remaining process are removed before we try to restart or migrate the resource group to another server. This is because it may cause a problem if some processes are still running, for example, if a process has a file open in the Domino data directory, we might not be able to unmount the file system. This would prevent the resource group from migrating to another node because the volume group cannot be *vary off* when a file system in it is mounted. To be sure that all Domino resources are removed, the stop script must remove all processes, shared memory segments, and semaphores. The main reason for creating two notes users notes1 and notes2 to run each Domino instance is that this makes it possible to Remove processes, shared memory segments, and semaphores for one domino instance when both instances are running on the same server. Figure 106 on page 197 shows a sample script to stop the Domino server started by user notes1 with the data directory in /server1.

```
#!/bin/ksh
#
#Shell script to stop Domino server in a HACMP environment
#02022000 IBM ITSO Austin,
#
PATH=/server1:$PATH
export PATH

echo  $LOGNAME | grep notes
if [ $? != 0 ]
then
        echo "not a accepted Domino user ! "
        exit 1
fi

/opt/lotus/bin/server  -q &
sleep 120

for i in ‘ps -ef | grep notes1 |grep -v ksh | grep -v grep | grep -v 'ps -ef' | \
awk '{ print $2 }' ‘

do
#        echo "removing process  " $i
kill -9 $i
done

for i in `ipcs | grep notes1 | awk '/^m / {print $2}'`
do
#       echo "removing Shared Memory " $i
        ipcrm -m $i
done

for i in `ipcs | grep notes1 | awk '/^s / {print $2}'`
do
#        echo "removing Semaphore " $i
       ipcrm -s $i
done

# cd /server1
# /opt/lotus/bin/nsd -kill
```

Figure 106. Domino server stop script

In a normal situation, the Domino server would stop within one minute. However, try it out on the actual installation and set the value in the stop script to the double value and then remove any other remaining resources.

12.7 Check HACMP takeover

After making the start and stop script and other HACMP configurations, check that the system is taking over and starting HACMP and the Domino server correctly.

Note

Define the TCP/IP address used by the Domino server in notes.ini file.

Because Domino is running in two instances on the same server, in case of a takeover, we must define on which TCP/IP address the Domino server is listening. In the default Domino configuration, the server opens any TCP/IP port for service. This implies that if we have two Domino servers running on the same AIX server with two IP service addresses, both Domino servers would be listening on both ports—with unpredictable results. To check which ports are used by a server, start a Domino administrator screen and select **server -> status**. In the command field, issue the `Show port tcpip` command. The output from the show command with a default setting is shown in Figure 108. The `*` means that the server is listening on all ports. This is not usable in a partitioned server and we must restrict the ports where the Domino server is serving.

```
Notes Session    Local Address          Foreign  Address
 0A0B0001       9.3.187.191:1352       9.3.240.101:3041
 0A130002       9.3.187.191:1352            *:*
```

Figure 107. Restricted TCP/IP ports used by the Domino server

A detailed description on how to restrict the port usage can be found in the Domino database *Domino 5 Administration Help,* which can be found on any Domino server. To limit the TCP/IP port usage for the Domino server serving Notes clients, add the following line to the notes.ini file on each server:

On server bmw, the IP address is 9.3.187.191:

- TCPIP_TcpIpAddress=0,9.3.187.191:1352

On server vw, the IP address is 9.3.187.187:

- TCPIP_TcpIpAddress=0,9.3.187.187:1352

From the Domino server console, it can be verified which IP addresses will be served by a server. Issue the `show port tcpip` command. The output, shown in

Figure 108 on page 199, is for a server that only listens for requests on one IP address.

```
Notes Session    Local Address          Foreign  Address
0A0B0001                    *:1352       9.3.240.101:3041
0A130002                    *:1352                      *:*
```

Figure 108. Unrestricted TCP/IP ports used by the Domino server

Figure 108 shows the port status from a server with no restrictions. The asterisk (*) means that the server is serving incoming requests on any IP address. This setting cannot be used with a partitioned Domino server where more Domino servers are running on the same system.

12.8 Making Domino highly available

In the previous sections, we described how to install Domino in an HACMP environment. HACMP, however, is monitoring status of the HW, not the status of an application such as Domino. As a user, we care about the Domino service, not the underlying HW. This means that we must monitor the Domino status and take appropriate actions if the service becomes unavailable.

The goal for the application monitor program is to check the service delivered by the Domino server in a stable cluster. It is not the goal of the program to handle any HW problems. HACMP is made for that, and it is not likely that we can improve the availability with a small application test program.

To monitor the status, we need a program that can run on any node and determine whether or not the Domino server is serving clients. The functions we need are:

- Check that the Domino data directory is available. This means that the resource group is active and the HACMP cluster is up. Only reports errors if the cluster is stable and there is no answer from the Domino server.
- From the local AIX server, poll the Domino server and check for valid results.
- The program should operate in quiet mode, with no print out unless specified. Errors are detected from return code 0=OK 0!= error.
- Log mode where status is written with a time stamp.

To create such a program, we need a C compiler for AIX, the Lotus C API Toolkit for Domino and Notes, Release 5.0.2, which can be downloaded from:

```
http://www.lotus.com/home.nsf/welcome/developernetwork
```

The programming documentation comes with the C API downloadable images as a Domino database.

We also need the manual *HACMP V4.3 AIX: Programming Client Applications*, SC23-4282.
The source code for our test program hadomtest.c is listed in Appendix B, “Domino HACMP script and programs” on page 277.

The program in our example takes three arguments, which are:

<server name>	The Domino server we want to test.
<database filename>	A database name we will try to open.
[debug] Optional	A debug value to control how detailed of output we want.

The program will try to contact the Domino server and ask for the database title for the Domino database with the file name <database filename>. The program returns a value 0, or not 0, for a successful connection or a unsuccessful connection, respectively. This program can be used in a shell script to check the server. To fully understand the principle of the application monitor program, consider the configuration shown in Figure 109 on page 201.

We have the two servers, bmw and vw, with shared disks. We have defined two resource groups in HACMP, bmdominovg and vwdominovg. The bmwdominovg resource group contains the file system /server1, the IP label bmw, and Domino server bmw in Domino partition one. Vwdominovg resource group contains the file system /server2, the IP label vw and Domino server vw in Domino partition two. The test program is started together with the Domino server and runs continuously (forever). The program is supposed to check which resource groups are local, and then check that the local resource groups have Domino servers running. The program logic is:

- Initial wait until the Domino servers has started.
- While continuously running (forever):
  - If /server1 is mounted, check that server bmw is running.
  - If not, restart server bmw.
  - If /server2 is mounted, check that server vw is running.
  - If not, restart server vw.
  - Sleep two minutes before trying again.
- Done.

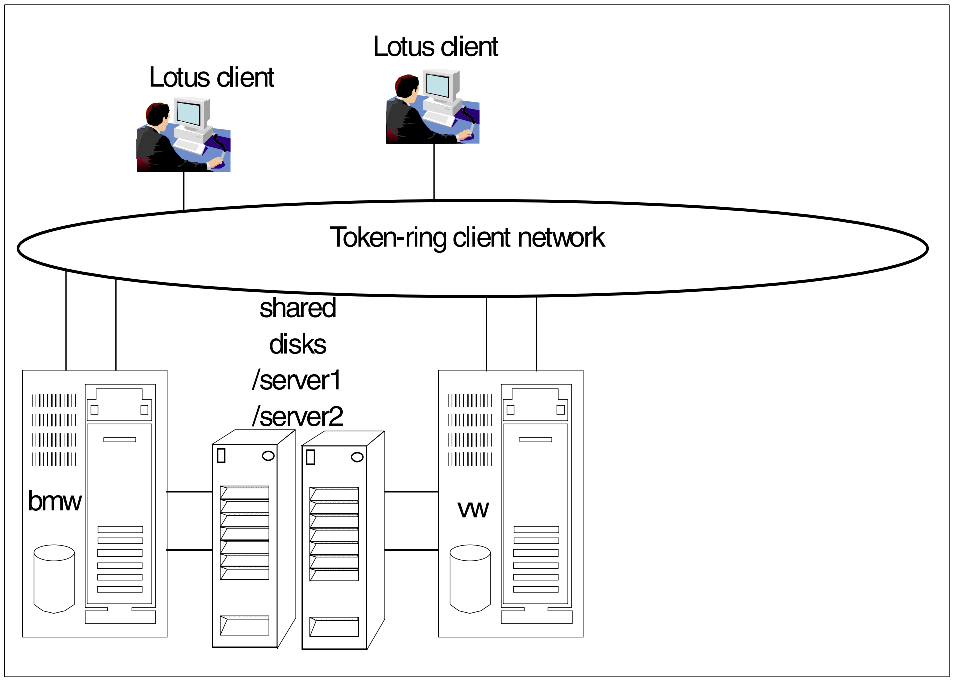


Figure 109. HACMP 2-node cluster with Domino servers

The programs used to start and stop the Domino server in the resource group consists of call_start_srvX, start_serverX, call_stop_srvX, stop_serverX, hadomtest, and check_server, where X is replaced with 1 for user notes1 and 2 for user notes2. All programs with index 1 are placed in /u/notes1 at both system bmw and vw and programs with index 2 are placed in /u/notes2 on both systems. The start procedure is outlined in Figure 110 on page 202. Call_start_srvX is the HACMP application server start script. Call_start_srvX call two other scripts—start_serverX and check_server. Check_server and call_start_srvX will run as root, because they are started by HACMP. start_serverX will run as user notes1 or notes2, because it's very important that the Domino server is not running as root but as the designated user.

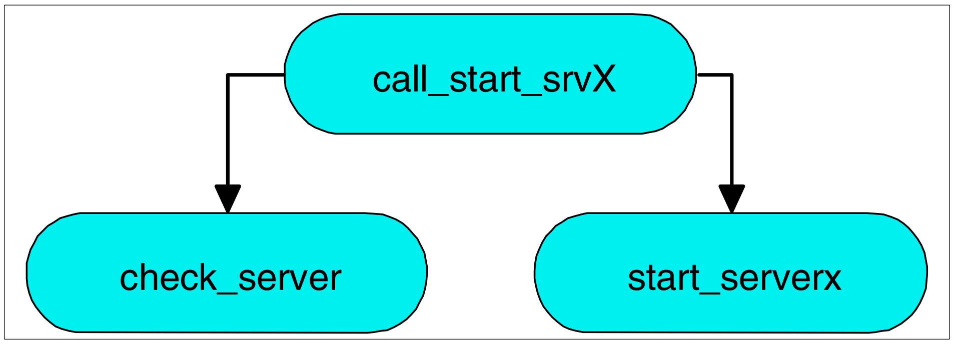


Figure 110. Domino start procedure

The procedure to check the Domino server is somewhat more complicated. It is described in Figure 111 and consists of a indefinite loop. For each Domino server instance notes1 and notes2, the program checks if the data file system is mounted locally. If it is not, we assume that the resource group is active on the other node and we do not do any further checks. If the file system is mounted, however, we assume that the resource group is active and we try to connect to the server with the test program hadomtest. If the server fails to respond to the test program, we close the Domino server to make sure it is properly done and then restart the server.

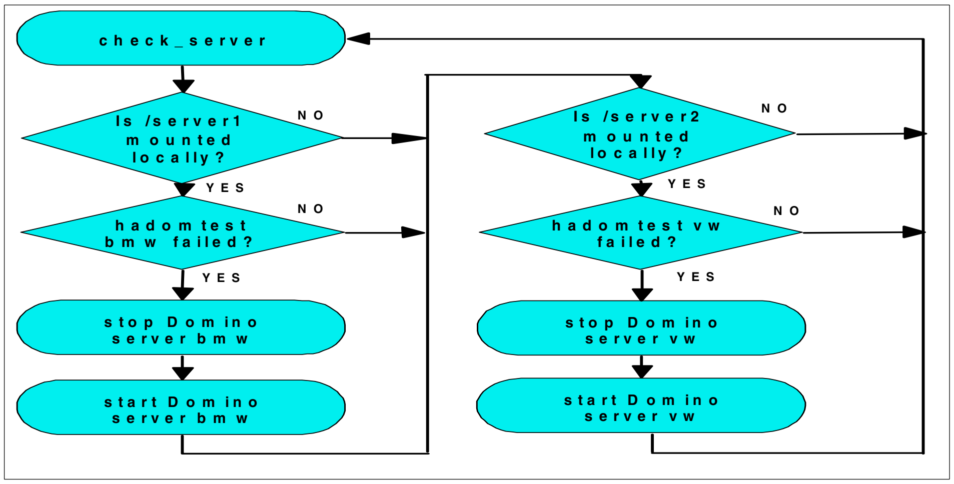


Figure 111. Domino check procedure

When the operator wants to stop the server, it should be controlled by HACMP, or at least by calling call_stop_srvX, which is the HACMP application server stop script. At first, the check_server script has to be stopped to prevent the server from restarting immediately, and then we can stop the Domino server. The stop procedure is shown in Figure 112.

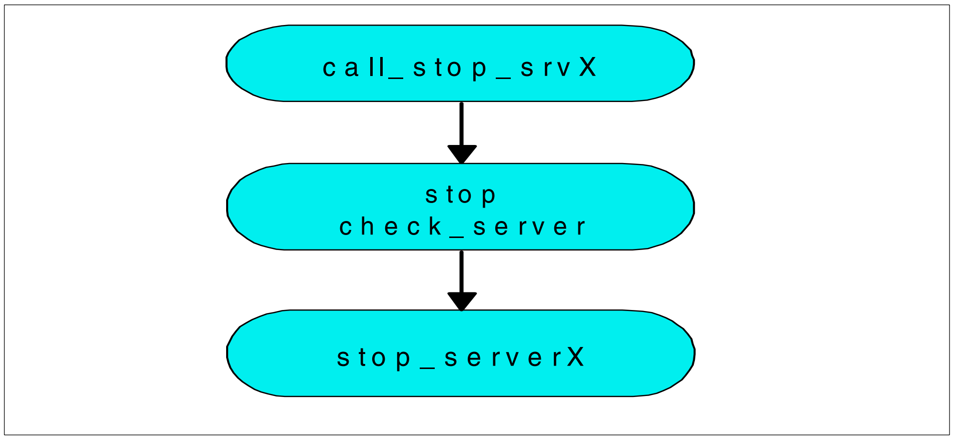


Figure 112. Procedure for stopping the Domino server

The sources for all the programs are listed in Appendix B, “Domino HACMP script and programs” on page 277.

12.9 Domino client functionality

In the described environment, we have successfully connected to the two servers from workstations. We have also connected to both servers, bmw and vw, independently when they are running on one AIX server. We have also been able to open a document on a workstation served by one server (vw), and then halt the AIX server and let a takeover occur and then save the document on the Domino server now running on another AIX system. This means that we have a nonstop application server seen from the workstation.

Chapter 13. AIX Error Notification and device monitoring

In a highly available environment, we should check the status of all of our resources to avoid causing the cluster resources to fall over. HACMP monitors the network adapters and the availability of other cluster nodes. Monitoring devices, such as disk adapters, disk drives, and adapter cards, requires additional customizing and scripting. The right tool to do this is the *AIX Error Notification* facility.

13.1 AIX Error Notification tool

AIX Error Notification allows you to run your own shell scripts or programs automatically in response to specified errors appearing in the AIX error log. Hardware and software errors and operator messages are logged in the AIX error log. Each time an error is logged in the system error log, the error notification daemon checks the defined *notification objects,* and if the error log entry matches the selection criteria, it runs the notify methods.

The notification objects are error conditions including a wide range of criteria to define. These errors can be any kind of software or hardware errors known by your AIX system. Also, you can add notification objects for your application's error log entry if the application uses the AIX system error log.

The notify methods can be any kind of shell scripts or commands. They are usually shell scripts that include e-mail or pager notifications to the system administrators and appropriate steps to recover from the error or shut down the cluster.

AIX already contains some predefined notification objects. The active error notification object definitions are stored in the *errnotify* ODM class.

13.1.1 Automatic error notification

You can configure predefined notification objects for hard, non-recoverable errors for the following resources on all cluster nodes with the HACMP automatic error notification utility:

- All disks in the rootvg volume group
- All disks defined in HACMP resource group as file systems, volume groups, raw logical volumes, and raw disks
- All disk adapters used by HACMP resources
- SP Switch adapter

HACMP assigns the notification objects with one of the following notify events:

- **cl_failover:** Sends an error message to the console and hacmp.out including the error log entry label and the failing device name. The script also mails the error message to the root. It stops the cluster gracefully with takeover on the failing node. This notify method is assigned with non-recoverable hardware errors that can cause the cluster resources to fail over.
- **cl_logerror:** Logs the error label and failing device name in the hacmp.out file and also sends e-mail to the root for non-serious errors.

You can add your own functions to the cl_failover and cl_logerror scripts as well as change the notification objects.

13.1.2 Configuring automatic error notification

Before you configure the automatic error notification, you must have a valid HACMP topology and resource configuration.

Note

Automatic error notification should be configured only when the cluster is not running.

To configure automatic error notification, perform the following steps:

1. Be sure the cluster is not running.
2. Start system management for HACMP:

   ```
   smit hacmp
   ```

3. Select **RAS Support -> Error Notification -> Configure Automatic Error Notification**.
   - To setup the automatic error notification, select **Add Error Notify Methods for Cluster Resources**.
   - Delete Error Notify Methods for Cluster Resources removes HACMP's predefined notification objects from the active configuration.
   - Select **List Error Notify Methods for Cluster Resources** to list the configured automatic error notification objects.

You can use the `/usr/es/sbin/cluster/sbin/cl_errnot` (`/usr/sbin/cluster/sbin/cl_errnot` in the Classic version) command to add, list, or delete automatic error notification. A sample output of the

`/usr/es/sbin/cluster/sbin/cl_errnot -l` command can be seen on Figure 113.

```
bmw# /usr/es/sbin/cluster/sbin/cl_errnot -l
vw:
vw: HACMP Resource              Error Notify Method
vw:
vw: hdisk0             /usr/es/sbin/cluster/diag/cl_failover
vw: hdisk1             /usr/es/sbin/cluster/diag/cl_failover
vw: hdisk2             /usr/es/sbin/cluster/diag/cl_failover
vw: scsi0              /usr/es/sbin/cluster/diag/cl_failover
vw: ssa0               /usr/es/sbin/cluster/diag/cl_failover
bmw:
bmw: HACMP Resource             Error Notify Method
bmw:
bmw: hdisk0            /usr/es/sbin/cluster/diag/cl_failover
bmw: hdisk1            /usr/es/sbin/cluster/diag/cl_failover
bmw: hdisk2            /usr/es/sbin/cluster/diag/cl_failover
bmw: scsi1             /usr/es/sbin/cluster/diag/cl_failover
bmw: ssa0              /usr/es/sbin/cluster/diag/cl_failover
```

Figure 113. Sample output of the /usr/es/sbin/cluster/sbin/vl_errnot -l command

If you make any changes in the hardware or cluster configuration, you may need to reconfigure the automatic error notification. Later, you can edit or remove each automatic error notification object as normal error notification objects in the SMIT **RAS Support -> Error Notification** menu on each node.

13.1.3 Defining a customized error notification object

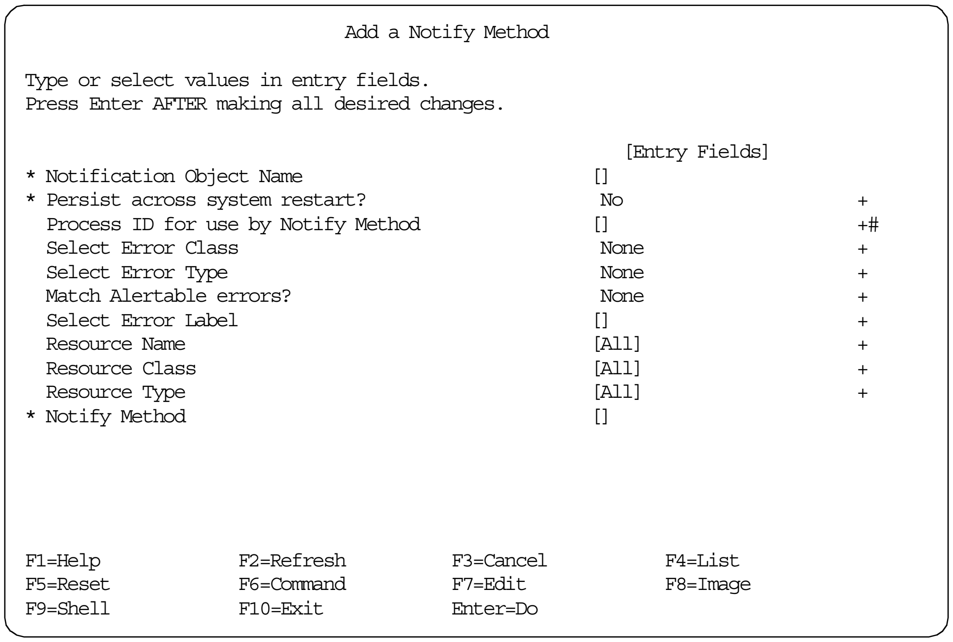
HACMP provides SMIT menus to configure AIX Error Notification. You must define the notification objects manually on each node. To define an error notification object and notify method, complete the following steps:

1. Start system management for the HACMP:

   ```
   smit hacmp
   ```

2. Select **RAS Support -> Error Notification -> Add a Notify Method**.

   You can see the Add a Notify Method SMIT panel in Figure 114 on page 208.



```
                              Add a Notify Method

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
* Notification Object Name                            []
* Persist across system restart?                       No                     +
  Process ID for use by Notify Method                 []                     +#
  Select Error Class                                   None                   +
  Select Error Type                                    None                   +
  Match Alertable errors?                              None                   +
  Select Error Label                                  []                      +
  Resource Name                                       [All]                   +
  Resource Class                                      [All]                   +
  Resource Type                                       [All]                   +
* Notify Method                                       []






F1=Help              F2=Refresh           F3=Cancel            F4=List
F5=Reset             F6=Command           F7=Edit              F8=Image
F9=Shell             F10=Exit             Enter=Do
```

Figure 114. Add a Notify Method SMIT panel

3. Enter the values for the following fields:

Notification Object Name	Enter a user-defined name that identifies the error notification object.
Persist across system restart?	Set this field to `Yes` if you want to use this notification object persistently. Set this field to `No` if you want to use this object until the next reboot.
Process ID for use by Notify Method	You can specify a process ID for the notify method to use or you can leave it blank. Objects that have a process ID specified should have the Persist across system restart field set to `No`.
Select Error Class	Choose the appropriate error class. Valid values are `None`: No error class match, `All`: All error classes, `Hardware`: Hardware error class, `Software`: Software error class, and `Errlogger`: Operator notifications, messages from the `errlogger` command.

Select error type	Identify the severity of error log entries. Valid values are `None`: No entry type to match, `All`: Match all error types, `PEND`: Impending loss of availability, `PERM`: Permanent, `PERF`: Performance degradation, `TEMP`: Temporary, and `UNKN`: Unknown error type.
Match Alertable Errors?	This field is provided for use by network management applications alert agents. Chose `None` to ignore this entry. Valid values are `All`: Match all alertable errors, `TRUE`: Matches alertable errors, and `FALSE`: Matches non-alertable errors.
Select Error Label	Select an error label associated with the accurate error identifier from the /usr/include/sys/errids.h file. Press **F4** for a listing. If your application supports the AIX system error log, you can specify the application-specific error label.
Resource Name	The name of the failing resource. For the hardware error class, this is the device name. For the software class, this is the name of the failing executable. Specify **All** to match all error labels.
Resource Class	For the hardware error class, the resource class is the device class. It is not applicable for software errors. Specify `All` to match all resources classes.
Resource Type	Enter the device type by which a resource is known in devices object for hardware error class. Specify `All` to match all resource types.
Notify Method	Enter the full-path name of the executable file, shell script, or command to be run whenever an error is logged that matches the defined criteria. You can pass the following variables to the executable: $1 Error log sequence number $2 Error ID $3 Error class $4 Error type $5 Alert flag $6 Resource name of the failing device $7 Resource type of the failing device $8 Resource class of the failing device $9 Error log entry label

4. Press **Enter** to create the notification object.

You have now defined an error notification object with the corresponding notify method. Later, you can modify or delete this objects in the **RAS Support -> Error Notification** SMIT panel**.**

13.1.4 Testing the error notification objects

HACMP can emulate error log entries. To test the notification objects:

1. Start system management for the HACMP:

   ```
   smit hacmp
   ```

2. Select **RAS Support -> Error Notification -> Emulate Error Log Entry**.
3. Select the desired notification object from the list.
4. Now, you can check the error label, notification object name, and notify method.
5. Press **Enter** to run the emulation.

The `/usr/es/sbin/cluster/utilities/err_convert` command creates the appropriate error log entry in the system error log, and the `errdemon` command automatically starts the notify method.

13.1.5 Example: Hardware error notification

The following example gives suggestions of how to monitor permanent hardware errors in an HACMP environment. We monitor the SSA disk array, and in case of SSA_DISK_ERR4 (unrecoverable hardware errors), the notification method sends an e-mail to the system administrator with the name of the disk and stops the cluster on the failing node.

The hardware error notification object for this example:

Notification Object Name	SSA_DISK
Persist across system restart?	Yes
Process ID for use by Notify Method	
Select Error Class	Hardware
Select Error Type	PERM
Match Alertable errors?	None
Select Error Label	SSA_DISK_ERR4
Resource Name	All
Resource Class	disk

Notification Object Name	SSA_DISK
Resource Type	All
Notify Method	/usr/es/sbin/cluster/events/ssa_disk_error $6

Notice that we put parameter $6 after the executable name. This passes the failing device name to the script.

The /usr/es/sbin/clsuter/events/ssa_disk_error script is our notify method for this error condition:

```
#!/bin/ksh
# First create the body of the letter in a temporary file
# Date of the error
date >/tmp/message
# The message
echo SSA disk error!>>/tmp/message
echo The failing disk is $1 >>/tmp/message
echo Please check the failing disk immediately!>>/tmp/message
# Send the content of the temporary file to the sysadmin
mail -s "SSA disk error!" root@bmw.itsc.austin.ibm.com </tmp/message
# Stop the cluster graceful with takeover (HACMP/ES)
/usr/es/sbin/cluster/utilities/clstop -y -N -gr
# Stop the cluster graceful with takeover (HACMP Classic)
# /usr/sbin/cluster/utilities/clstop -y -N -gr
```

13.1.6 Example: Software error notification

In this example, we monitor the PGSP_KILL error. This is a very serious error. It indicates that the system is out of paging space, and it cannot free any virtual memory. If the error notification daemon receives this error log event, the notify method halts the system immediately.

The software notification object for this example:

Notification Object Name	OUT_OF_MEMORY
Persist across system restart?	Yes
Process ID for use by Notify Method	
Select Error Class	Software
Select Error Type	None
Match Alertable errors?	None
Select Error Label	PGSP_KILL

Notification Object Name	OUT_OF_MEMORY
Resource Name	All
Resource Class	All
Resource Type	All
Notify Method	/usr/sbin/halt -q

13.1.7 Example: Application error notification

If the application supports AIX system error logs and puts meaningful error messages into the error log, then you can use the notification objects to react to application errors. Contact your application provider to decide if your application uses the AIX system error log and to provide a complete list of error log entries.

In this example, the application gives an error log entry in case of an unexpected execution stop, such as divide by zero. You can see the error log entry in Figure 115.

```
LABEL:           APP_UNEXP_STOP
IDENTIFIER:      BC99FA80
Date/Time:       Thu Jan 27 10:50:25
Sequence Number: 47
Machine Id:      000011655C00
Node Id:         bmw
Class:           S
Type:            UNKN
Resource Name:   myapp
Description
SOFTWARE
Probable Causes
MYAPPLICATION
Failure Causes
MYAPPLICATION
Recommended Actions
Restart the application
Detail Data
Divide by zero
```

Figure 115. Sample application error log entry

The AIX Error Notification object for the application error log entry shown in Figure 115:

Notification Object Name	APPLICATION_ERROR
Persist across system restart?	Yes

Notification Object Name	APPLICATION_ERROR
Process ID for use by Notify Method	
Select Error Class	Software
Select Error Type	None
Match Alertable errors?	None
Select Error Label	APP_UNEXP_STOP
Resource Name	All
Resource Class	All
Resource Type	All
Notify Method	/usr/myapp/restart_application

Notice that the APP_UNEXP_STOP label is not included in the SMIT-generated error label list.

For more information about the AIX system error log and user-defined error messages, see “Error Logging Overview” in *AIX Version 4.3 Problem Solving Guide and Reference*, SC23-4123.

13.2 Device monitoring

A few devices are not supported directly by either HACMP or by AIX Error Notification. These devices sometimes becomes one of the most important hardware components of the cluster. For a PPP dial-up or bank card authorization server, the availability of the multiport serial adapter is as important as the availability of the network adapters. For monitoring such a device, we need additional customizing and scripting.

The basic idea for device monitoring is to create a script that periodically checks the status or the availability of the device. Most of the device driver packages provide commands for status checking. The `lsdev` command can be used as a basic device monitoring tool but usually it is not a good choice. We suggest you use device-specific commands or try to examine the log files of the device driver if it is present. We suggest running the monitoring script from the crontab. Usually a 5 to 10 minutes interval is adequate, but this depends highly on the device and how sensitive the cluster must be for the unavailability of the monitored device. Also, create a simple locking mechanism to avoid running multiple instances of the script in case the script failed and hung. Always send the error messages in the hacmp.out file. In case of any error, save detailed diagnostic information to make it easier for further examination.

A possible skeleton for a device monitoring script is as follows:

1. Check if the cluster is running or not. If the cluster is not running, then terminate.
2. Ensure that there is no other running instance of the script. Use a simple lock file for this purpose. If the lock file is present, send an error message and terminate.
3. Create the lock file.
4. Check the device status or availability. If the resource is OK, then terminate and remove the lock file.
5. In case of any problem, notify the system administrator by mail, send a detailed error message into the hacmp.out file, and do the appropriate steps to recover from the device error. For example, try to restart the device driver or stop the cluster node.
6. Remove the lock file.

13.2.1 Example: Monitoring the ARTIC 960 X.25 card

The X.25 WAN connection is commonly used in banking. The banking applications often use their own protocol but TCP/IP or SNA over the X.25 lines. The X.25 SNA links can be easily monitored by HACMP and SNA communication servers. You can also check the TCP/IP communication over X.25 links with standard TCP/IP commands, such as `ping` or `netstat`. But what about the application-specific protocols?
In this example, we examine how to monitor the availability of the ARTIC 960 Multiport X.25 adapter and X.25 links with non-standard communication protocol. Neither the ARTIC 960 Multiport X.25 adapter nor AIXLink X.25 software packages support an AIX system error log. Therefore, we cannot use the AIX Error Notification facility to check the status of the X.25 communication links. The only possibility is to write our device monitoring script based on the `x25status` command. The output of the `x25status` command under normal circumstances is shown in Figure 116 on page 215.

```
bmw# x25status
********************************************************************
X.25 LINK STATUS for bmw
********************************************************************
                                        ACTIVE  ACTIVE  ACTIVE
PORT            PACKET STATE            SVCs    PVCs    LISTENS
sx25a0          Connected               3       0       0
sx25a1          Connected               1       0       1

********************************************************************
X.25 LISTEN STATUS for bmw
********************************************************************
PORT            CALLING-ADDRESS      CALLED-ADDRESS       CUD
sx25a1          *                    15011120500*         *
```

Figure 116. Output of x25status when the X.25 subsystem is OK

The command shows that there are three active outgoing switched virtual circuits (SVC) on the sx25a0 device and one incoming SVC on sx25a1.

The output of the `x25status` command when the lines are physically disconnected is shown in Figure 117.

```
bmw# x25status
********************************************************************
X.25 LINK STATUS for bmw
********************************************************************
                                        ACTIVE  ACTIVE  ACTIVE
PORT            PACKET STATE            SVCs    PVCs    LISTENS
sx25a0          Disconnected            0       0       0
sx25a1          Disconnected            0       0       0

********************************************************************
X.25 LISTEN STATUS for bmw
********************************************************************
PORT            CALLING-ADDRESS      CALLED-ADDRESS       CUD
```

Figure 117. Output of the x25status when the X.25 subsystem failed

In Figure 117 the `x25status` command shows that there are no active incoming or outgoing SVCs and that the links are disconnected. The cause of the disconnected status may be a software error or a physical error in the X.25 adapter, router, or line.

Our X.25 device monitoring script examines the output of x25status. It checks the number of active SVCs on the sx25a0 device and the number of active listens on sx25a1. These numbers are highlighted in the above examples. In our example, there are three active SVCs on the sx25a0 interface and one active listen on the sx25a1 interface. If the X.25 subsystem failed, the number of SVCs on the sx25a0 interface is 0 and there is not an active listen on the sx25a1 port.

In case of an error, the script stops the cluster node with takeover and sends the error message and the output of the `x25status` into the hacmp.out file. The script runs from the root's crontab every five minutes. It uses a simple lock file to avoid running in multiple instances if the script is hung. Also, if the HACMP is down, it simply terminates and does not perform the check. The following sample X.25 device monitoring script implements the above mentioned check and actions:

```
#!/bin/ksh
##########################################################################
# X.25 device monitoring sample script
##########################################################################
# The log file for the script
LOG=/tmp/hacmp.out
# Temporary file to store the output of the x25status command
OUT=/tmp/x25.out
# The ip label of the node's service adapter
SERVICE=bmw
# Is HACMP running?
netstat -i|grep -w $SERVICE>/dev/null 2>&1
if [ "$?" != "0" ]
then
    # The HACMP is not running, nothing to check
    exit 0
fi
# HACMP is up
# Check the lock file to avoid running the script
# in multiple instance
if [ -e "/etc/locks/x25_test.lock" ]
then
    # /etc/locks/x25_test.lock file is present, another instance of this
    # script is currently running. This is a real bad news.
    echo "**********************************************">>$LOG
    date>>$LOG
    echo "The X.25 device monitoring script hang. Please check">>$LOG
    echo "the X.25 devices, the x25_test script and">>$LOG
    echo "the /etc/locks/x25_test.lock file!">>$LOG
    echo "If everything seems like OK just remove the lock file!">>$LOG
    exit 1
fi
# Creating the lock file
```

```
touch /etc/locks/x25_test.lock
# Run the x25status command and store its output in a temporary file
/usr/bin/x25status>$OUT 2>&1
# Check the exit code of x25status
if [ "$?" != "0" ]
then
    # The exit code is not zero, something is wrong with the X.25 subsystem
    echo "************************************************">>$LOG
    date>>$LOG
    echo "The X.25 subsystem or the x25status does not work!">>$LOG
    # Put some useful data into the logfile
    lsdev -C|grep x25 >>$LOG 2>&1
    cat $OUT >>$LOG 2>&1
    # Remove the lock file and start the takeover
    rm  /etc/locks/x25_test.lock
    /usr/es/sbin/cluster/utilities/clstop -y -N -gr
    exit 2
fi
# Check the sx25a0 interface
# Count the number of active SVCs on sx25a0
SVCS=`grep sx25a0 $OUT | grep onnected | awk {' print $3 '}`
if [ "$SVCS" != "3" ]
then
    # The number of active SVC does not match. Some of the links are
    # disconnected
    echo "************************************************">>$LOG
    date>>$LOG
    echo "The number of active SVCs is not 3!">>$LOG
    echo "Some of the links are broken!">>$LOG
    # Put some useful data into the logfile
    lsdev -C|grep x25 >>$LOG 2>&1
    cat $OUT >>$LOG 2>&1
   # Remove the lock file and start the takeover
    rm  /etc/locks/x25_test.lock
    /usr/es/sbin/cluster/utilities/clstop -y -N -gr
    exit 2
fi
# sx25a0 OK
# Check sx25a1 interface
# Count the number of active listens on sx25a1
SVCS=`grep sx25a1 $OUT | grep Connected | awk {' print $5 '}`
if [ "$SVCS" != "1" ]
then
    # The number of active listens does not match. Some of the links are
    # disconnected
    echo "************************************************">>$LOG
    date>>$LOG
    echo "The X.25 listener process stopped or the link is broken!">>$LOG
    # Put some useful data into the logfile
    lsdev -C|grep x25 >>$LOG 2>&1
    cat $OUT >>$LOG 2>&1
    # Remove the lock file and start the takeover
    rm  /etc/locks/x25_test.lock
    /usr/es/sbin/cluster/utilities/clstop -y -N -gr
    exit 2
fi
# sx25a1 OK
# The X.25 subsystem is ok. Remove the lock file and exit.
rm  /etc/locks/x25_test.lock
exit 0
```

This sample script checks the X.25 subsystem if the cluster is up; otherwise, it terminates and does not perform any action. First, it checks its lock file to ensure that there is no other running instance of this script. Then, the script runs the x25status command and examines its output. If the exit code of the x25status command is not zero, or the number of active SVCs and listens are not equal to the appropriate numbers, the device monitoring scripts notifies the root user about the X.25 error via e-mail and starts an HACMP takeover. If the X.25 subsystem is all right, the script removes the lock file and terminates.

13.2.2 Example: Monitoring the 128-port async adapter card

The 128-port async adapter with Remote Asynchronous Nodes (RAN) boxes is widely used for modem pools. Its device driver does not support AIX Error Notification. IBM provides libraries for programing the RANs, but it is a very time consuming way to create a monitoring tool. The `/usr/lbin/tty/mon-cxma` command is an interactive utility to check the status of the RANs and the ports. We cannot use interactive programs in a script, so we need to find other solutions to monitor the async adapters.

The easiest way to test a 128-port serial adapter’s functionality is by using it with a loopback device attached between its two ports. We send data through the loopback cable from one port to another. If we can receive the data on another node, the RANs and the async card is OK. In case of any error, the read process hangs (actually, it is waiting for data for an endless time). You can easily implement this monitoring method, but it requires at least two empty ports in the RANs.

The loopback device is a standard RS-232 cross wired (null modem) serial cable connected between two ports of the RAN as shown in Figure 118 on page 219.

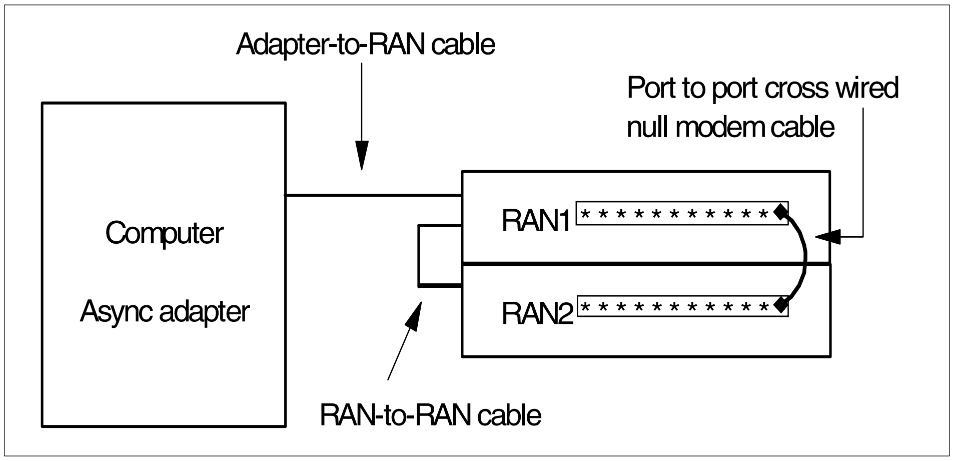


Figure 118. RAN cabling

Our example async tty monitoring script uses this method to test the availability of the RANs. It sends a short message to the first tty and reads from the other one via the loopback device. In case of the failure of any component of the 128 port async adapter or the RANs, read from the tty hangs. With a simple lock file, we can check if the pervious run of the script is terminated or not. If the script did not terminate, it indicates the failure of the async adapter. The script runs from the crontab in every five minutes. If the cluster is not running, it simply terminates.

Perform the following steps to set up the async adapter device monitoring:

1. Install the cross wired null modem cables between two RAN ports.
2. Start system management for the TTY device:
   ```
   smit tty
   ```
3. Select **Add a TTY**.
4. Select **tty rs232 Asynchronous Terminal** for TTY Type.
5. Select the appropriate async adapter card for the Parent Adapter.
6. On both ports, set up the following values:
   - PORT number: The appropriate port number on the RAN
   - Enable LOGIN: Disable
   - BAUD rate: 9600
   - PARITY: None

- BITS per character: 8
- Number of STOP BITS:1

Leave the other settings on the default values.

7. Add the following two lines to the /etc/uucp/Devices file:

   ```
   Direct ttyX - Any direct
   Direct ttyY - Any direct
   ```

   Here `X` and `Y` are the appropriate TTY numbers for the test ports.

8. Test the communication between the ports:

   ```
   echo "hello world" >/dev/ttyX &
   cat </dev/ttyY
   ```

   If everything is all right, you receive back the `"hello world"` message.

9. Run the following sample 128-port async adapter monitoring script from the crontab every five minutes.

The sample 128-port async adapter device monitoring script:

```
#!/bin/ksh
########################################################################
# 128 port async adapter and RAN device monitor script
########################################################################
# The log file
LOG=/tmp/hacmp.out
# The node's service address IP label
SERVICE=bmw
# The test tty ports
TTY_READ=tty18
TTY_WRITE=tty34
# First, check is HACMP running?
netstat -i|grep -w $SERVICE>/dev/null 2>&1
if [ "$?" != "0" ]
then
    # The cluster is down, terminate
    exit 0
fi
# Check the lock file. If it is present then the pervious instance of this
# script hung. This indicates the failure of the async adapter or the RAN.
if [ -e "/etc/locks/tty_test.lock" ]
then
    # The lock file is present: RAN error!
    echo "***********************************************">>$LOG
    date>>$LOG
    echo "128 port async adapter or RAN error!!!">>$LOG
    echo "Please check the async adapter, RAN devices and cables!">>$LOG
```

```
    echo "If there is no problem with them please remove the">>$LOG
    echo "/etc/locks/tty_test.lock lock file.">>$LOG
    # Terminate
    exit 1
fi
# Create the lock file
touch /etc/locks/tty_test.lock
# The test:
# Send date from TTY_WRITE to TTY_READ
# Send a message from TTY_WRITE
echo "hello world" >/dev/$TTY_WRITE &
# Read back the message from TTY_READ. In case of any error the script
# will hang here.
cat </dev/$TTY_READ >/dev/null 2>&1
# If the cat command terminates then everything is OK
# Remove the lock file
rm  /etc/locks/tty_test.lock
exit 0
```

First, the script checks the HACMP. If the cluster is down, the script terminates. Otherwise, it checks the presence of the lock file from a pervious run of this script. If the lock file is present, this indicates a problem with the 128-port async adapter or RAN devices and the script notifies the system administrator.

If the pervious run of this device monitoring script was successful, then the script creates its own lock file and executes the RAN test: it sends a few bytes through the RAN-attached loopback device and tries to catch it. If the RANs are bad, the script cannot receive the message and it hangs without removing its lock file. Finally, if everything goes all right, the script removes its lock file and terminates.

Chapter 14. Dependencies between applications

The flexibility in HACMP configurations allows multiple applications to be integrated into cluster resource groups. These applications can be independent of each other, or they can be dependent on one another. This chapter describes implementing a cluster with multiple applications that depend on one another.

In addition, we discuss event customization to achieve additional tasks not provided by HACMP.

14.1 Defining application dependencies

When working with more than one application that depend on each other, a careful plan is required to integrate the application seamlessly into an HACMP cluster.

14.1.1 Simple multi-application failover

Figure 119 on page 224 shows a sample cluster with one resource group that is comprised of five different applications. The standby node is an idle node with no applications. We are using the sample cluster to help explain application dependencies.

14.1.1.1 Understanding an environment

A system administrator should understand how applications work in his or her environment. As a system administrator, you should have enough understanding of the requirements of each application in order to outline their dependencies. Once the dependencies are confirmed, draw a flow diagram of the sequence of the application startup, application shutdown, and application takeover.

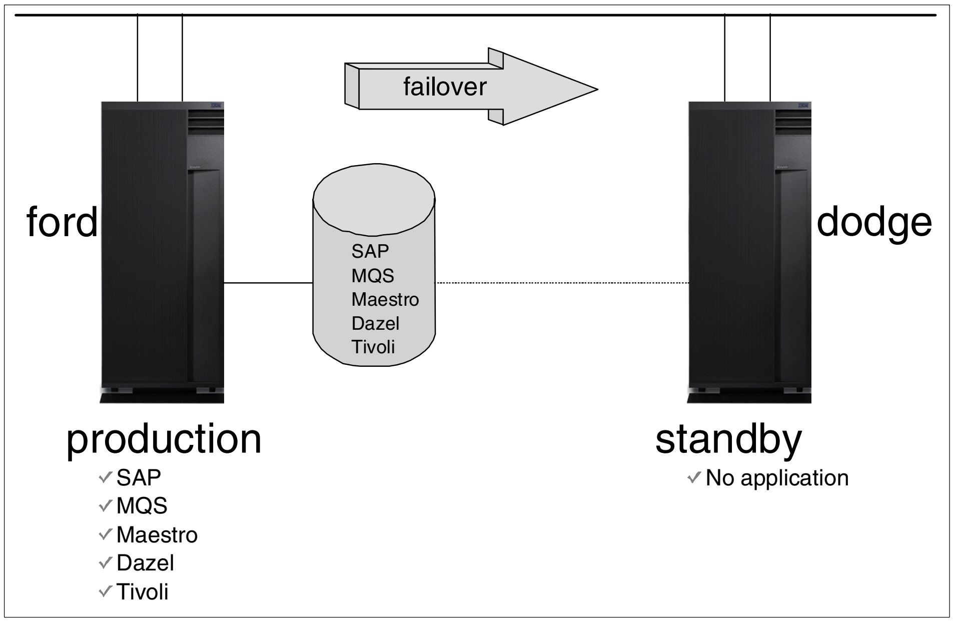


Figure 119. Application failover to an idle standby node

The flow diagram in Figure 120 shows the relationship between applications in the cluster shown in Figure 119. The flow diagram shows that SAP must be started first, followed by MQSeries, Maestro, and Dazel, respectively. Tivoli can be started anytime after MQSeries is started, and it is not dependent on Maestro and Dazel.

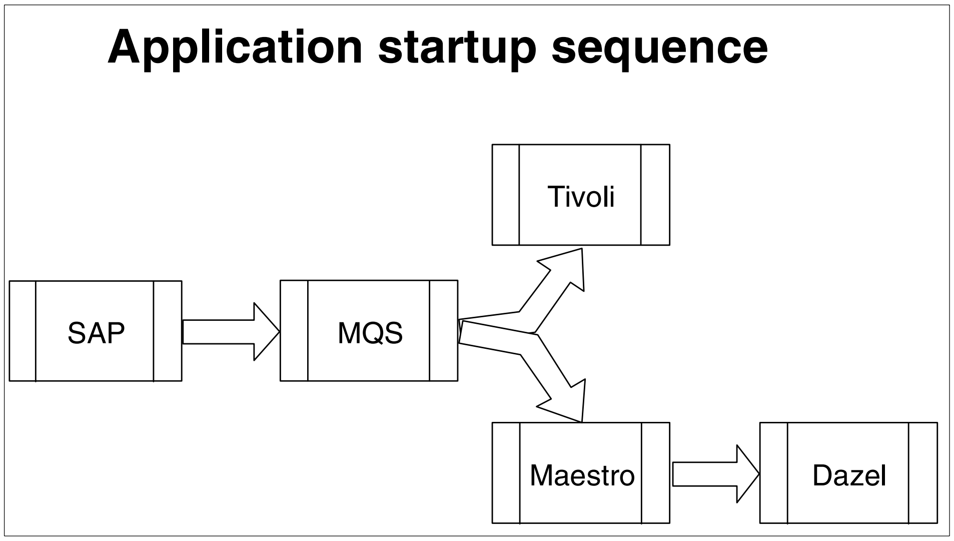


Figure 120. Flow chart of an application sequence

14.1.1.2 Preparing application startup scripts

The simplest way to control application depencies is to define a single application server to manage all the applications. That means, there is only one application startup script and one shutdown script.

Each individual application has its own startup and shutdown scripts. The individual applications scripts are called by the main scripts defined in the application server. Therefore, the application can be started in any sequence without major modifications of the script.

The individual application scrips are listed in Table 18.

Table 18. Individual application scripts

Application	Start Script	Stop Script
SAP	start_sap	stop_sap
MQSeries	start_mqs	stop_mqs
Maestro	start_maestro	stop_maestro
Dazel	start_dazel	stop_dazel
Tivoli	start_tivoli	stop_tivoli

Using the information in Table 18, a simple startup script can be developed. Figure 121 shows a sample of a simple script that can be used to start the applications:

```
#!/bin/sh
/usr/sbin/cluster/local/start_sap
/usr/sbin/cluster/local/start_mqs
/usr/sbin/cluster/local/start_maestro
/usr/sbin/cluster/local/start_dazel
/usr/sbin/cluster/local/start_tivoli
```

Figure 121. A simple multi-application startup script

The script shown in Figure 121 starts each application one by one. The next application only starts after the previous application startup is complete. The next application starts even if the previous application fails. If you want to start the next application only when the previous application has successfully started, you have to build in error checking in the script. A sample is shown in Figure 122.

```
#!/bin/sh
/usr/sbin/cluster/local/start_sap
if [ $? -eq 0 ]
then
        /usr/sbin/cluster/local/start_mqs
        if [ $? -eq 0 ]
        then
                /usr/sbin/cluster/local/start_tivoli
                /usr/sbin/cluster/local/start_maestro
                if [ $? -eq 0 ]
                then
                        /usr/sbin/cluster/local/start_dazel
                else
                        exit 3
                fi
        else
                exit 2
        fi
else
        exit 1
fi
```

Figure 122. A multi-application startup script with error checking

The previous two examples are considered simple scripts that run in a cluster that consists of a production node and an idle standby node.

14.1.2 Multi-application failover with additional dependencies

When additional criteria need to be considered during an application failover, the application failover becomes more complex. The sample cluster shown in Figure 123 has the same application on the production node as the previous cluster in Figure 119 on page 224. However, the standby node is running the Oracle Test and Development instance.

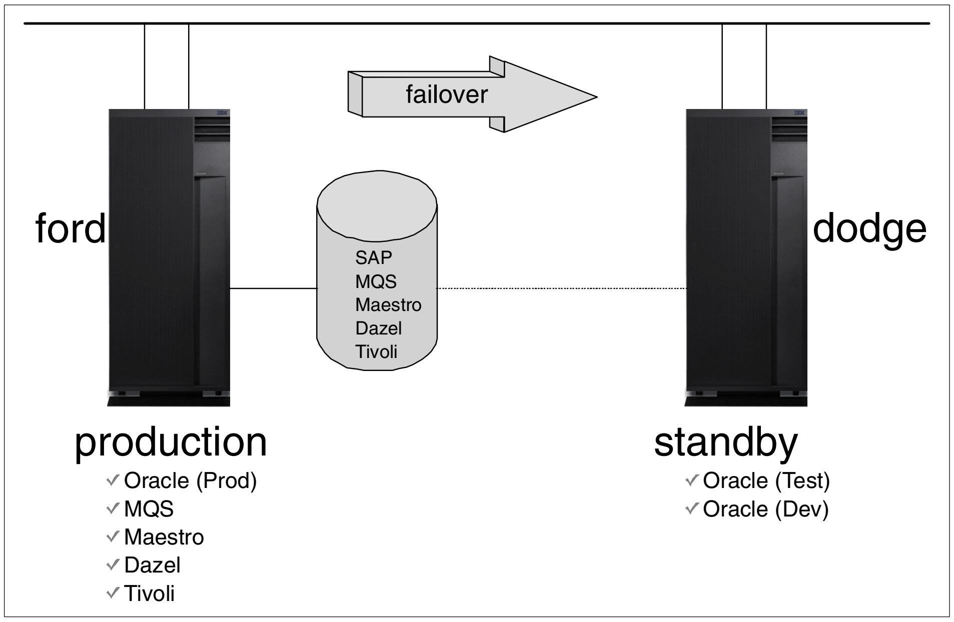


Figure 123. Application failover to a standby node with local applications

In this scenario, there are decisions to be made. What should we do with the applications running on the standby node? Since the development and test environment is not critical to a business, the applications on the standby node should be stopped. The following are reasons why it should be stopped:

- To provide more CPU resources to the production applications
- To ensure that the production environment and the development do not coexist in the same node

Configuring the application scripts can be tricky, and comprehensive tests should be performed to ensure that the configuration works in every possible situation. We go through several possible solutions to this scenario.

14.1.2.1 Integrating all scripts into an application server

The easiest method is to launch and stop all applications using application servers. However, the startup script and the shutdown script are written with some environment checking.

The startup script behaves differently on the production node and the standby node. The script should detect the node it runs on and execute different tasks on different nodes. By doing so, the same copy of a startup and a shutdown script can be used on both nodes.

A sample startup script is shown in Figure 124.

```
#!/bin/sh

CLPATH=/usr/sbin/cluster/utilities

if [ $CLPATH/get_local_nodename = dodge ]
then
        /usr/sbin/cluster/local/stop_ora_tstdev
fi
/usr/sbin/cluster/local/start_oracle_prd
if [ $? -eq 0 ]
        then
                /usr/sbin/cluster/local/start_mqs
                if [ $? -eq 0 ]
                then
                        /usr/sbin/cluster/local/start_tivoli
                        /usr/sbin/cluster/local/start_maestro
                        if [ $? -eq 0 ]
                        then
                                /usr/sbin/cluster/local/start_dazel
                        else
                                exit 3
                        fi
                else
                        exit 2
                fi
else
        exit 1
fi
```

Figure 124. A sample start script with node dependency

The startup script first checks whether the node is dodge or ford. In a normal startup on ford, the script starts the application normally. However, during failover, the script detects that it is running on dodge. Therefore, it stops the local applications before starting the failed over applications.

The stop script must also be written so that when the ford rejoins the cluster and acquires all resources, the Oracle development instances are restarted on the standby node. Figure 125 on page 229 shows a sample stop script.

```
#!/bin/sh

CLPATH=/usr/sbin/cluster/utilities

/usr/sbin/cluster/local/stop_oracle_prd
/usr/sbin/cluster/local/stop_mqs
/usr/sbin/cluster/local/stop_tivoli
/usr/sbin/cluster/local/stop_maestro
/usr/sbin/cluster/local/stop_dazel

if [ $CLPATH/get_local_nodename = dodge ]
then
        /usr/sbin/cluster/local/start_ora_tstdev
fi
```

Figure 125. A sample stop script with node dependency

The stop script in Figure 125 stops all production applications and restarts the local applications on the standby node. When the script runs on ford, it only shuts down all production applications.

14.1.2.2 Using event customization to control local applications

The startup and shutdown of applications on the standby server can also be controlled using event management. The customization of the applications are discussed in 14.3, “Customizing event scripts” on page 233.

14.2 Sequence of cluster events

An event is a change of status within a cluster. When the Cluster Manager detects a change in cluster status, it executes the designated script to handle the event and initiates any user-defined customized processing.

A different set of events is triggered in a different phase of a cluster activity. In order to customize a cluster event, we should look into the sequence of events in different situations.

14.2.1 A first node joins a cluster

The following are the sequence of events triggered locally when a first node joins a cluster:

1. node_up starts
2. node_up_local starts
3. acquire_service_addr starts
4. acquire_aconn_service starts

5. swap_aconn_protocols starts
6. swap_aconn_protocols ends
7. acquire_aconn_service ends
8. acquire_service_addr ends
9. get_disk_vg_fs starts
10. get_disk_vg_fs ends
11. node_up_local ends
12. node_up ends
13. node_up_complete starts
14. node_up_local_complete starts
15. start_server starts
16. start_server ends
17. node_up_local_complete ends
18. node_up_local_complete starts
19. node_up_local_complete ends
20. node_up_complete ends

No events are triggered on the remote node.

14.2.2 A second node joins a cluster or a failed node rejoins

When a second node joins a cluster, the same events are triggered locally. However, there are some events triggered on the remote node. The events are as follows:

1. node_up starts
2. node_up_remote starts
3. stop_server starts
4. stop_server ends
5. release_vg_fs starts
6. release_vg_fs ends
7. release_takeover_addr starts
8. release_takeover_addr ends
9. node_up_remote ends
10. node_up ends

11.node_up_complete starts
12.node_up_remote_complete starts
13.node_up_remote_complete ends
14.node_up_complete ends

The same sequence occurs when a failed node rejoins the cluster.

14.2.3 A node exits a cluster gracefully

When a node is shuts down gracefully, the following events are triggered on the node itself:

1. node_down starts
2. node_down_local starts
3. stop_server starts
4. stop_server ends
5. release_vg_fs starts
6. release_vg_fs ends
7. release_service_addr starts
8. release_service_addr ends
9. node_down_local ends
10. node_down_local starts
11. release_vg_fs starts
12. release_vg_fs ends
13. node_down_local ends
14. node_down ends
15. node_down_complete starts
16. node_down_local_complete starts
17. node_down_local_complete ends
18. node_down_local_complete starts
19. node_down_local_complete ends
20. node_down_complete ends

On the remote node:

1. node_down starts

2. node_down_remote starts
3. node_down_remote ends
4. node_down ends
5. node_down_complete starts
6. node_down_remote_complete starts
7. node_down_remote_complete ends
8. node_down_complete ends
9. network_down starts
10. network_down ends
11. network_down_complete starts
12. network_down_complete ends

No events are triggered on the remote node.

14.2.4 A node exits a cluster in takeover mode

When a cluster node exits the cluster in a takeover mode, the same events are triggered as when the node exits the cluster gracefully.

However, on the remote node, the takeover process is performed by the following events:

1. node_down starts
2. node_down_remote starts
3. acquire_takeover_addr starts
4. acquire_takeover_addr ends
5. get_disk_vg_fs starts
6. get_disk_vg_fs ends
7. node_down_remote ends
8. node_down ends
9. node_down_complete starts
10. node_down_remote_complete starts
11. start_server starts
12. start_server ends
13. node_down_remote_complete ends

14.node_down_complete ends

The sequence of events is presented to help us in customizing the cluster events. By having some information about how the events are organized, customizing the events scripts will be easier. We can pick the best event to customize since we know exactly when it is triggered by the Cluster Manager.

14.3 Customizing event scripts

The HACMP for AIX system is event-driven. An event is a change of status within a cluster. When the Cluster Manager detects a change in cluster status, it executes the designated script to handle the event and initiates any user-defined customized processing.

To configure cluster events, you indicate the script that handles the event and any additional processing that should accompany an event.

The HACMP for AIX system provides a default script for each predefined cluster event and subevent. You can tailor event processing for your environment by adding multiple pre- or post-event processing of your own design. HACMP 4.3 also allows you to add, change, and remove customized cluster events.

Note

Even though the event scripts are written in shell scripts, it is not advisable to modify the scripts directly because it complicates maintenance and administration. Therefore, you should use event customization to add pre- and post-events scripts.

HACMP event customization is used when adding additional tasks during the execution of the standard HACMP events. For example, when using an IP aliasing in an SAP environment, the acquire_service_addr event is customized so that the post event start_aliasing is executed. Several other events are customized to ensure that the IP aliasing works transparently with SAPR/3.

14.3.1 Sample event customization

Local applications are applications that are not configured in an HACMP resource group and, therefore, are not controlled by the cluster. The following sample of event customization refers to the application environment shown in Figure 123 on page 227.

The task is to customize the event scripts to perform the following:

1. Stop Oracle development and test during failover.
2. Start Oracle development and test during reintegration.

First of all, we should identify the event that should be customized. The previous section lists all the events triggered during failover and reintegration.

The Oracle application on the standby node should be stopped before the production application is started. Therefore, we can customize the start_server event so that the pre-event executes the Oracle stop script.

To restart Oracle on the standby node after reintegration, the node_up_remote should be customized by adding a post-event.

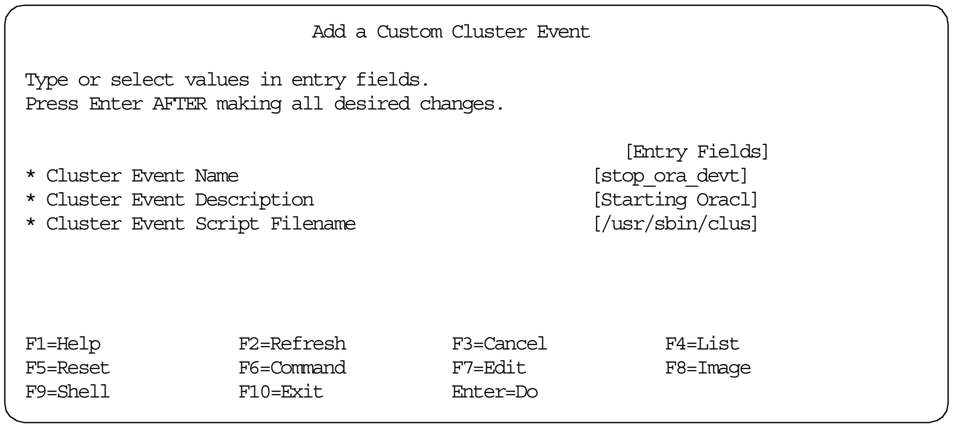
The following are steps to customize the start_server event:

1. Use the following command:

   ```
   smitty hacmp
   ```

2. Select **Cluster Configuration -> Cluster Resources -> Cluster Events -> Define Custom Cluster Events -> Add a Custom Cluster Event**.

   The panel shown in Figure 126 appears.

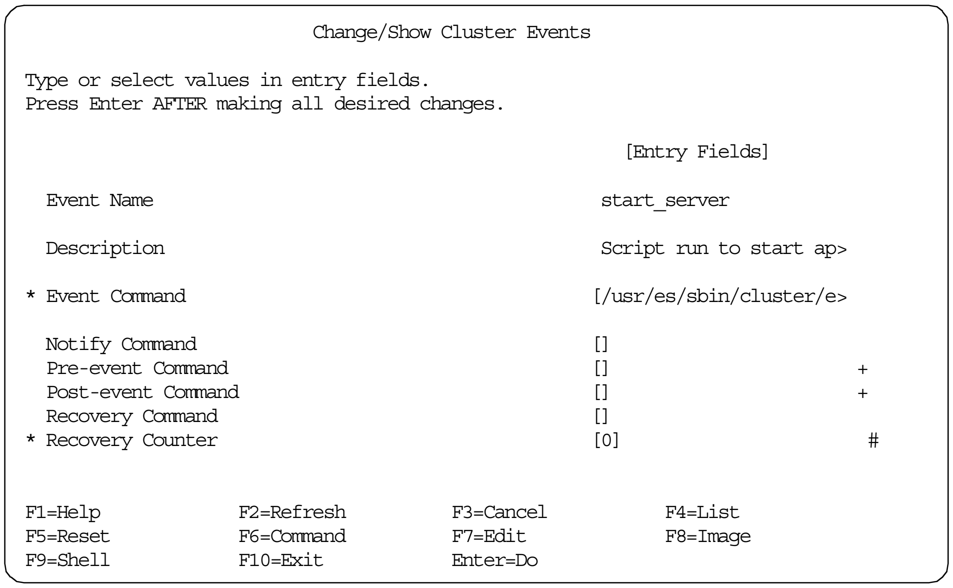


```
                              Add a Custom Cluster Event

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                          [Entry Fields]
* Cluster Event Name                                   [stop_ora_devt]
* Cluster Event Description                            [Starting Oracl]
* Cluster Event Script Filename                        [/usr/sbin/clus]




F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 126. Adding a custom event

3. Fill in the Cluster Event Name field as `stop_ora_devt` and the Cluster Event Script Filename as `/usr/sbin/cluster/stop_ora_sby`. Press **Enter** to commit the selection.

4. Repeat step 1 through step 3. Fill in the Cluster Event Name field as `stop_ora_devt` and the Cluster Event Script Filename as `/usr/sbin/cluster/stop_ora_tstdev`. Press **Enter** to commit the selection.
5. Press **F3** twice until you get to the Cluster Events screen. Select **Change/Show Cluster Events -> start_server**. The panel shown in Figure 127 appears.



```
                              Change/Show Cluster Events

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                          [Entry Fields]

  Event Name                                            start_server

  Description                                           Script run to start ap>

* Event Command                                        [/usr/es/sbin/cluster/e>

  Notify Command                                       []
  Pre-event Command                                    []                    +
  Post-event Command                                   []                    +
  Recovery Command                                     []
* Recovery Counter                                     [0]                    #


F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 127. Changing a start_server cluster event

6. In the Pre-event Command field, press **F4** and select **start_ora_devt**. Press **Enter** to commit the changes.

The event is successfully changed. Repeat steps 1 through 6 with the following information to customize the node_up_remote event:

- Cluster Event Name: `start_ora_devt`
- Cluster Event Script Filename: `/usr/sbin/cluster/local/start_ora_sby`

Copy the scripts defined in the custom events to both nodes. Sample scripts are shown in Figure 128 on page 236 and Figure 129 on page 236.

```
#!/bin/sh

CLPATH=/usr/sbin/cluster/utilities

if [ $CLPATH/get_local_nodename = dodge ]
then
        /usr/sbin/cluster/local/stop_ora_tstdev
fi
```

Figure 128. Contents of /usr/sbin/cluster/local/stop_ora_sby

```
#!/bin/sh

CLPATH=/usr/sbin/cluster/utilities

if [ $CLPATH/get_local_nodename = dodge ]
then
        /usr/sbin/cluster/local/start_ora_tstdev
fi
```

Figure 129. Contents of /usr/sbin/cluster/local/start_ora_sby

7. Synchronize the cluster resources to propagate changes to all the cluster nodes.
8. Test the new events to ensure that they work accordingly.

Chapter 15. Writing a cluster aware application

A cluster aware application is an application that is capable of detecting the status of a cluster and behaving accordingly. The status can be acquired through a function call available in the clinfo application programming interface (API). When an application is able to retrieve the status of its data provider, it can also react to this information and wait for the resource to be available again before submitting a transaction if the resource is temporarily unavailable. Therefore, all transactions can be guaranteed to complete and are not lost if the server fails.

In this section, we create a sample application to demonstrate how the API provided with HACMP can be used to achieve high availability by acquiring the status of the server before obtaining any data, and if the server is unavailable, waiting until the server becomes available again. The application consists of separate modules. Each task is written as a separate program and all program modules are invoked from a shell script to make it easier for the reader to reuse part of it.

15.1 Application overview

The sample application that we are demonstrating is a simple application that checks the cluster status via the clinfo API tool and waits for the cluster availability in order to successfully commit a transaction. In Figure 130 on page 238, we sketch the basic idea of the implementation.

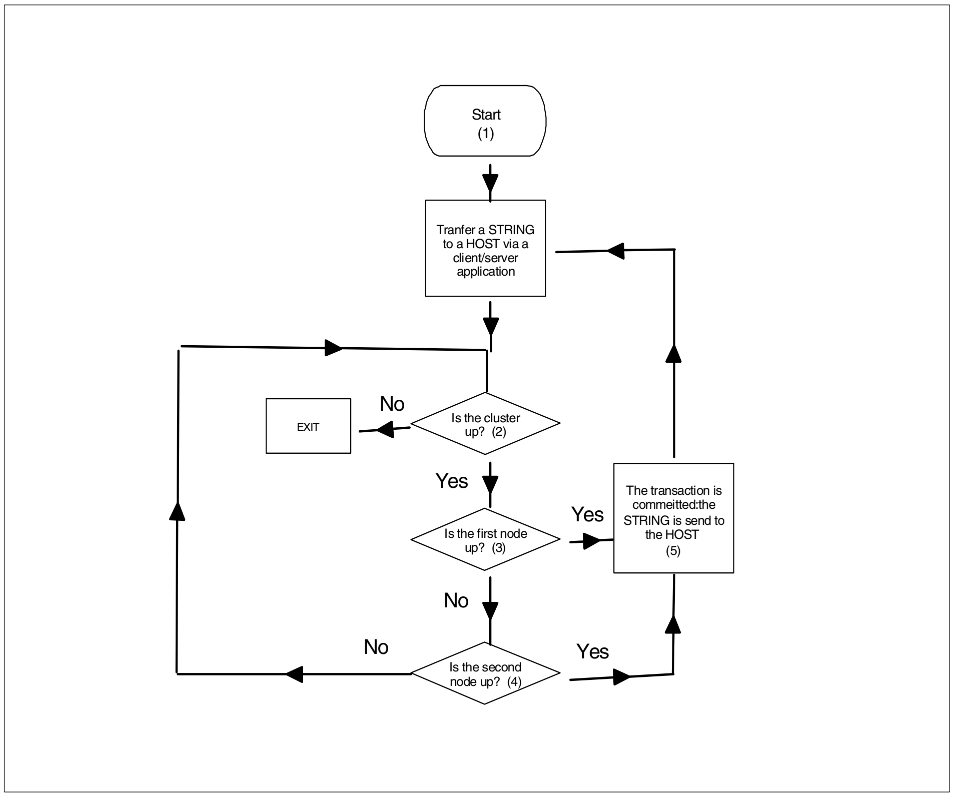

Start
(1)
Tranfer a STRING to a HOST via a client/server application
No
EXIT
Is the cluster up? (2)
Yes
Is the first node up? (3)
Yes
The transaction is commeitted:the STRING is send to the HOST (5)
No
No
Is the second node up? (4)
Yes

Figure 130. Application layout

We have a client/server application called TX that allows us to transfer strings. The txclient send a string to the HOST where the txserver runs as shown in Figure 130. The HOST is a cluster node and the application checks the cluster status in steps (2), (3), and (4) in Figure 130, and only if one of the two server nodes is running, is the string sent.

The components of the cluster aware application for this example are:

- A client/server application called TX, which is composed of the programs txserver and txclient.
- An HACMP/ES cluster with the nodes ford and dodge. The txserver runs on ford, and as ford fails, the txserver is taken over by the node dodge.
- A system called vw where the HACMP client has been installed.
- A C-program called clstatus, which is based on HACMP API and checks the cluster status.

- A shell script clp that runs on vw. The shell script checks the cluster status via the program clstatus and sends strings via txclient to a cluster node as soon as it is available.

The environment is summarized in Figure 131.

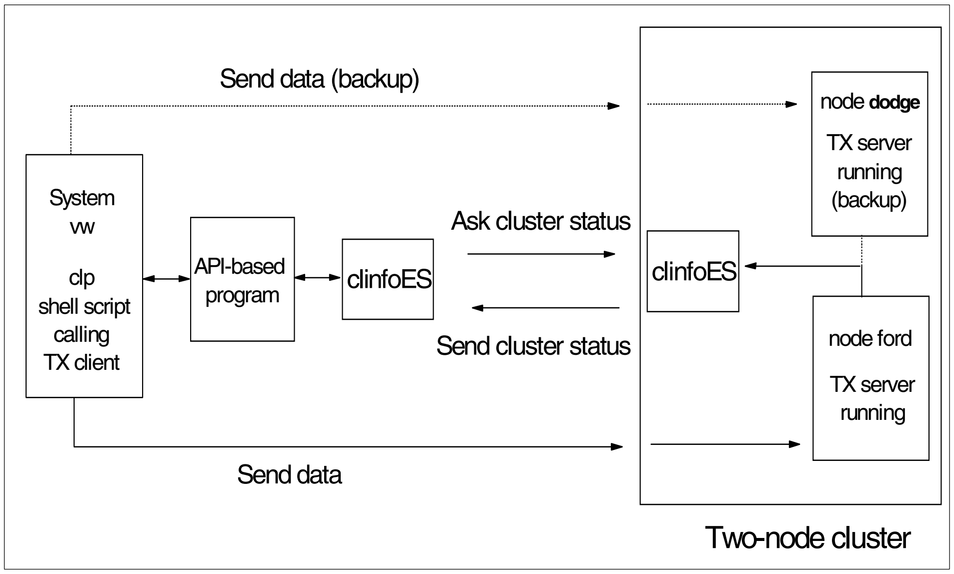


Figure 131. The environment

In the following sections, we describe the application more in detail and we show the application behavior during the cluster status changes.

15.1.1 TX application overview

Our sample application is client-server based. It is a simple transaction system: the client txclient sends data, the server txserver stores it in a file in the current directory and sends back an acknowledgment. The client and server applications use a standard TCP/IP connection.

You can see the program and data flow of the application in Figure 132 on page 240.

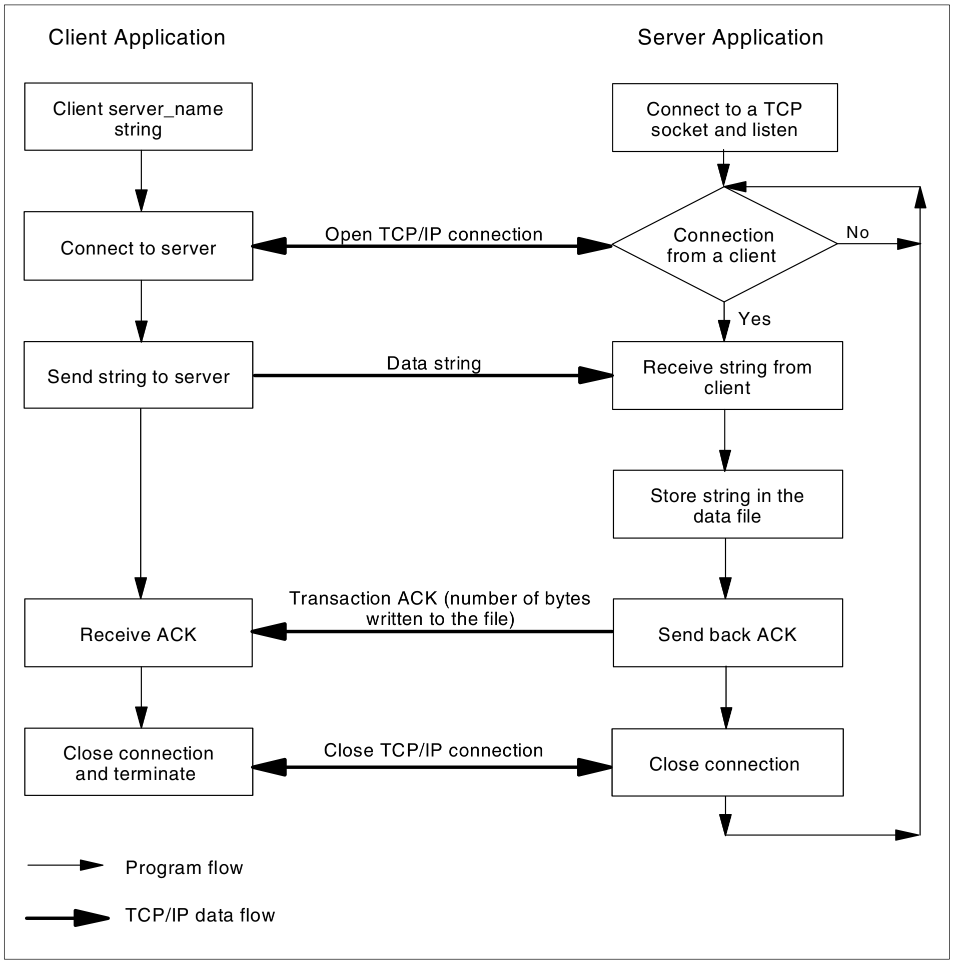


Figure 132. Flow chart of the sample client-server application

The TX server program runs as a background process on the server *host* and continuously listens for client connections. If a TX client connects, the client sends the data to the server. In our case, the data is a string given as command line arguments of the client. The server program receives the data and stores it in a file on a shared file system. Then, the server sends back an acknowledgment to the client. If its value is less then zero, it indicates a server I/O error. Otherwise, the acknowledgment is the number of bytes written out to the data file. Finally, the client and server close the TCP/IP connection.

We explain both server and client programs in the following sections.

15.1.2 The txclient environment

The client environment requires the following components:

- The file set cluster.es.client.rte, which is supplied with the HACMP/ES code
- The executable file clstatus, which is in the SG244498.ZIP file
- The executable file txclient, which is in the SG244498.ZIP file
- The shells script clp, which is in the SG244498.ZIP file

The txclient runs on the system vw where we installed the HACMP client file set cluster.es.client.rte 4.3.1.1 that contains the daemon clinfoES. We wrote a C-program, clstatus, that uses the HACMP API to check the cluster and node status. The clstatus program receives the cluster status from the client clinfoES that is configured to about inquire the cluster status on a remote host. We then made a script, clp, that sends a string (in particular, the time stamp of the transaction) via txclient from system vw to the system where the txserver is running. The script is cluster aware, that is, it checks the cluster status, and if a cluster node is up, it sends data. Eventually, the server acknowledges to the client that the string has been received as shown in Figure 133.

```
-----------------------------------------------------------------------------
                         TRANSACTION SERVICES CLIENT
-----------------------------------------------------------------------------
Checking cluster status
Checking cluster status
Host: ford, using port 24259
The string: Mon Feb 21 17:52:52 CST 2000
28 bytes was written to the strings.data file on the server
```

Figure 133. clp output of a transaction success

If the cluster node running txserver fails, the script detects the node failure and waits to send data until the backup node has taken over as shown in Figure 134 on page 242.

```
-----------------------------------------------------------------------------
                         TRANSACTION SERVICES CLIENT
-----------------------------------------------------------------------------
Checking cluster status
Checking cluster status
Server is not running!
Please verify that the application server is running on the node
-----------------------------------------------------------------------------
                         TRANSACTION SERVICES CLIENT
-----------------------------------------------------------------------------
Checking cluster status
Checking cluster status
Host: dodge, using port 24259
The string: Mon Feb 21 17:55:23 CST 2000
28 bytes was written to the strings.data file on the server
```

Figure 134. clp output during a takeover

As the former node rejoins the cluster, the script stops to commit the transactions until the txserver is running again as shown in Figure 135.

```
-----------------------------------------------------------------------------
                         TRANSACTION SERVICES CLIENT
-----------------------------------------------------------------------------
Checking cluster status
Checking cluster status
Host: ford, using port 24259
The string: Mon Feb 21 17:59:53 CST 2000
28 bytes was written to the strings.data file on the server
```

Figure 135. clp output as the node reintegration finishes

15.1.3 The txserver environment and the cluster environment

The server environment requires the following components:

- HACMP/ES code
- The executable file txserver, which is in the SG244498.ZIP
- The scripts demo_start.sh and demo_stop.sh, which are in the SG244498.ZIP

The txserver receives the data from txclient and stores it in a file. The txserver is installed in a high availability configuration—it runs on the node ford and, as ford fails, it starts to run on node dodge. The executable and the file (in our example, strings.data) where the data are stored is placed in the file system /sharedfs. This file system belongs to the volume group fordvg whose physical volumes are shared disks. An HACMP application server, demoapp, manages the start and stop of the txserver via the shell scripts demo_start.sh and demo_stop.sh. Eventually, the cascading resource group fordrg has the file system /sharedfs and the application server demoapp as resources. Ford is the primary node and dodge is the standby node in the resource group fordrg.

15.2 Transaction application modules

In this section, we describe each of the modules in the cluster aware application.

15.2.1 Client program

The client program opens a socket connection to the server. The server name and the data string are given by command line arguments. If the connection is successful, the client sends the data string to the server and waits for the acknowledgment. If the byte value of the acknowledgment is less than zero, it is treated as an error code from the server. Otherwise, the value of the acknowledgment is the number of bytes written out by the data file by the server. It must be equal to the length of the provided data string. Finally, the client closes the TCP/IP connection and terminates.

You can find the source code of the client application in Appendix C.1, “Socket send (TX client application)” on page 285. You can find the file txclient and txclient.c in the SG244498.ZIP file, which can be downloaded from `www.redbooks.ibm.com` following the link Associated material.

15.2.2 Cluster sensing program

Our sample cluster sensing program is based on the Clinfo C API. It checks the status of the cluster, the cluster node, and their interfaces. You can specify you cluster ID as a parameter for the test program.

The program’s prerequisite is the cluster.es.client (HACMP/ES) or cluster.client (HACMP Classic) file sets. You must configure and start clinfo on the client computer.

The test program uses the cl_getcluster API call to get the cluster status. It returns a data structure with the following information:

- Cluster ID
- Cluster name
- Primary node name
- Cluster status

The cl_getnodemap API call provides the following information for each node:

- Node name
- Node ID
- Cluster ID
- Node status
- Number of interfaces
- Name of each network interface
- Address of each interface
- Status of each interface

These two API calls provide all the necessary information about the cluster status. The cluster tester program displays this information.

You can find the clstatus and clstatus.c files in the SG244498.ZIP file, which can be downloaded from `www.redbooks.ibm.com` following the link Associated material.

15.2.3 Shell script

The main client application is written in a shell script that uses the executable modules to check the cluster status and submits transactions to the server. The flow of the shell script is shown in Figure 130 on page 238.

First, the script checks that the clinfo daemon is running on the client node. Since clinfo is a prerequisite for the client application to run, the script exits if the clinfo daemon is not active.

Before submitting any transactions to the server, the script calls cl_getcluster to check the cluster status. The command returns the status of all nodes in the cluster. If the primary node is available, the script calls the client program with the primary node as the target for the program.

When the primary node is not available, the transaction is automatically submitted to the backup server. The transaction repeats with the primary server as the target when it is available.

Refer to the sample script in Appendix C.4, “TX client application shell script” on page 295, or to the clp file in the SG244498.ZIP file, which can be downloaded from `www.redbooks.ibm.com` following the link Associated material.

15.2.4 Server application

The server process can be started as a daemon in the background. It opens a TCP/IP socket connection for listening. The server is capable of serving multiple clients at the same time. If a client connects to its TCP port, it starts the server function. The server receives data strings from the client, stores them in a file, and sends back the number of bytes written out. In case of a file I/O error, the server returns an error code.

For the server application source code, refer to Appendix C.2, “Socket receive (TX server application)” on page 287, or you can find the txserver and txserver.c files in the SG244498.ZIP file, which can be downloaded from `www.redbooks.ibm.com` following the link Associated material.

15.3 Conclusions

From this sample application, we can see that it is possible, with little effort, to make an application cluster aware. By adding this feature, we can add to the value of an application for the customer. Seen from the customer’s perspective, this small change in the client application makes a world of difference. Without the cluster awareness, the customer transaction fails, and the customer must roll back all their work and then try the transaction again later. With the addition of cluster status awareness, however, the client application senses the status of the cluster and can eventually report the status back to the user, and then wait for the cluster resource to become available again.

Chapter 16. Package solutions for HACMP

HACMP does not require the cluster nodes to be of equal size or capacity. An HACMP cluster is built from standard RS/6000 models with some redundant components. A different RS/6000 generation can be paired in an HACMP cluster. This has always been the strength of HACMP solutions.

A package of redundant hardware configurations and HACMP software provide a choice of bundled HACMP systems. Customers can choose from several different configurations based on their processing power requirements.

This chapter discusses the available packaged HACMP solutions, such as HA-H70, HA-S70, and HA-S80.

16.1 Standard configuration for a packaged cluster

The following is the minimum configuration for each cluster pair. Additional features can be added to the minimum configuration. More expansion racks may be required as you are adding additional storage and other peripheral devices.

16.1.1 RS/6000 HA-H70 cluster server

HA-H70 is the smallest cluster configuration among the HACMP packaged solutions. The standard configurations are as follows:

- Two IBM RS/6000 Enterprise Server Model H70; each server includes:
  - 1-way 340 MHz RS64 II microprocessor with 4 MB of level 2 cache
  - 512 MB of memory
  - Dual Integrated Ultra SCSI-2controllers
  - Dual 4.5 GB Ultra SCSI IPL boot disks
  - Dual advanced serial RAID PCI SSA disk adapters
  - Dual LAN adapters, either IBM 10/100 Mbps Ethernet PCI, or Gigabit Ethernet-SX PCI, SYSKonnect SK-Net
  - FDDI PCI, Turboways 155 ATM PCI
  - IBM PCI Token-Ring
  - Integrated 10/100 Ethernet PCI
  - Ethernet 10Base2 Transceiver

    - Redundant AC power supply
    - CD-ROM drive
    - 1.44 MB 3.5-inch diskette drive
    - HACMP Version 4.3
    - AIX Version 4.3.2 (1-2 users)
- IBM 7133 Model D40 Serial Disk System
    - Eight 4.5 GB or 9.1 GB or 18.2 GB SSA disks
    - Redundant data paths
    - Redundant AC power supply
- IBM 7014 Model S00 Rack
    - Redundant power distribution unit
- Cabling for H70 servers, 7133 disk system, and HACMP communications

The two-node cluster is packed into a single S00 rack. See Figure 136.

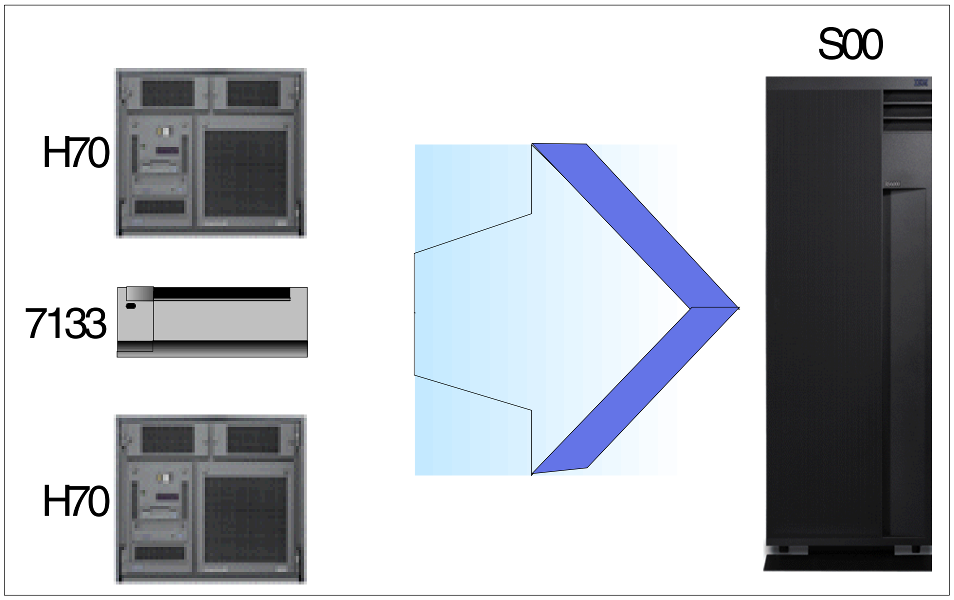


Figure 136. HA-H70 in a single rack

16.1.2 RS/6000 HA-S70 Advanced cluster server

The RS/6000 HA-S70 Advanced cluster server is comprised of the following components:

- Two IBM RS/6000 Enterprise Server S70 Advanced; each server includes:
  - One RS64 II processor, 4-way SMP 262 MHz, 8 MB L2 cache
  - 2 GB of memory
  - Dual, redundant Ultra SCSI, 6-pack hot-swap disk drives and backplanes
  - Dual, redundant 9.1 GB Ultra SCSI IPL boot disk drives
  - Dual, redundant PCI SSA multi-initiator/RAID EL adapters
  - Three, redundant PCI LAN adapters; choice of Ethernet, Token-Ring, FDDI, or ATM adapters
  - One asynchronous communications; choice of 8-port adapter, 128-port controller, serial-to-serial port cable, RJ-45 to DB-25 converter cable
  - Dual, redundant N+1 AC power
  - One I/O drawer with dual and redundant AC power, support processor, 32X SCSI-2 CD-ROM drive
  - HACMP Version 4.3 license (Enhanced Scalability feature, or Enhanced Scalability Concurrent Resource Manager feature, or High Availability Subsystem feature)
  - AIX Version 4.3 (1-2 users)
- Two IBM 7133 Model D40 Serial Disk System; each 7133 includes:
  - Eight 9.1 GB or 18.2 GB hot-swap SSA disk drives
  - Redundant data paths
  - Dual, redundant AC power supplies
- One IBM S70 Advanced I/O Rack
  - Redundant power distribution unit
- Cabling for HA-S70 Advanced servers, 7133 disk systems, and HACMP communications

See Figure 137 on page 250.

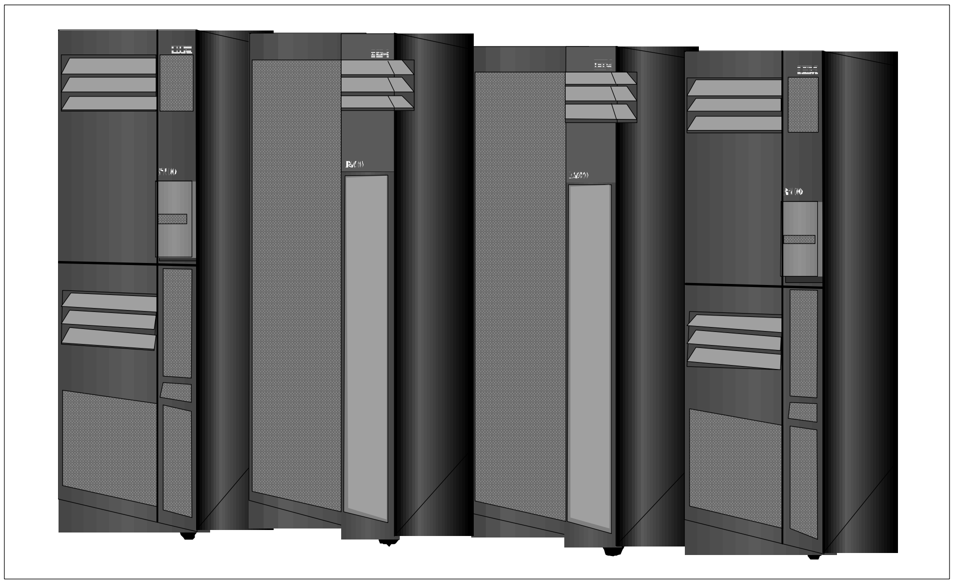


Figure 137. HA-S70 Advanced

16.1.3 RS/6000 HA-S80 cluster server

The HA-S80 has the following components:

- Two IBM RS/6000 Enterprise Server S80; each server includes:
  - One RS64 III processor, 6-way SMP 450 MHz, 8 MB L2 cache
  - 4 GB of memory
  - Dual, redundant Ultra SCSI, 6-pack hot-swap disk drives and backplanes
  - Dual, redundant 9.1 GB Ultra SCSI IPL boot disk drives
  - Dual, redundant PCI SSA multi-initiator/RAID EL adapters
  - Three, redundant PCI LAN adapters; choice of Ethernet, Token-Ring, FDDI, or ATM adapters
  - One asynchronous communications; choice of 8-port adapter, 128-port controller, serial-to-serial port cable, RJ-45 to DB-25 converter cable
  - Dual, redundant N+1 AC power
  - One I/O drawer with dual and redundant AC power, support processor, 32X SCSI-2 CD-ROM drive
  - HACMP/ES Version 4.3.1

    - AIX Version 4.3 (1-2 users)
- Two IBM 7133 Model D40 Serial Disk System; each 7133 includes:
    - Eight 9.1 GB or 18.2 GB hot-swap SSA disk drives
    - Redundant data paths
    - Dual, redundant AC power supplies
- One IBM S80 I/O Rack
    - Redundant power distribution unit
- Cabling for HA-S80 servers, 7133 disk systems, and HACMP communications

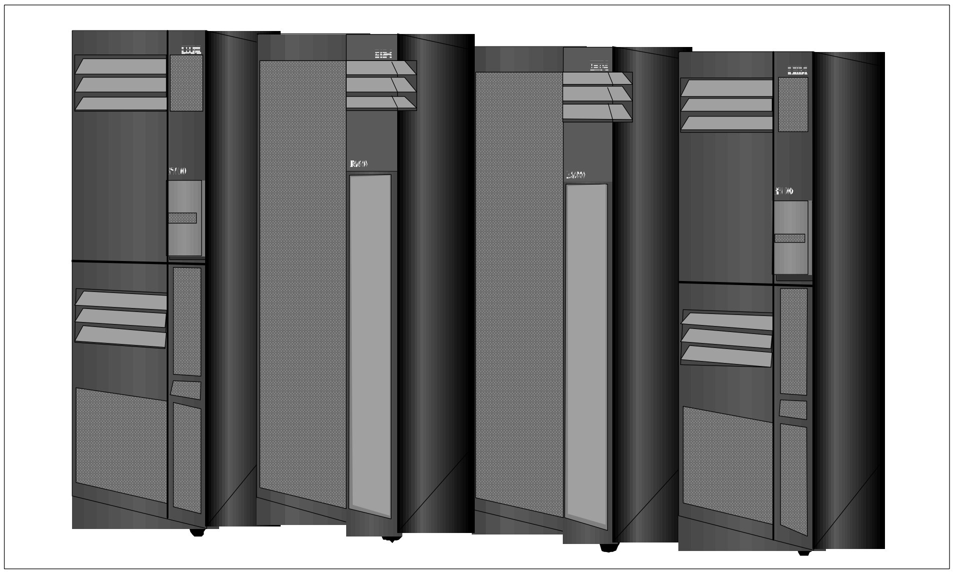


Figure 138. The RS/6000 HA-S80

16.2 Configuring the HACMP environment

An HACMP packaged system is a collection of redundant hardware with high availability software to provide a customer a highly available system. The system is not configured to be highly available. The solution delivered to a customer is HACMP ready, but they cannot work out of the box. The system should be configured as any regular HACMP implementation.

The following tasks need to be performed to complete the configuration and before HACMP can be started:

1. Complete the HACMP planning sheet.
2. Cable the SSA disks accordingly.
3. Connect and configure the serial connection between nodes.
4. Connect the two nodes to a LAN.
5. Configure all network adapters.
6. Install HACMP software on each node.
7. Configure shared disks.
8. Install applications.
9. Configure the cluster topology.
10. Configure the cluster resources.
11. Integrate applications into the resource group.
12. Test the failover.

Appendix A. SAP sample scripts

The following are the scripts used in customizing SAP R/3 to fit in the HACMP cluster environment.

A.1 Starting the DB server application

The script is modified to be used in an environment where a DB server is split for a CI server.

A.1.1 Start script: rc.startsap_prod_db

The following script starts the database:

```
#!/bin/ksh
#-----------------------------------------------------------------------
# Filename:rc.startsap_prod_db
# Path:/etc
# Node(s):HACMP-Server and Database-Server
# Info:Starts the Database and R/3
#-----------------------------------------------------------------------
# Remark:
# =======
# This script is prepared for SAP R/3 Version 3.0x
#
#-----------------------------------------------------------------------
# First set the environment and dump a timestamp
#-----------------------------------------------------------------------

. /usr/sbin/cluster/saptools/hacmpr3.profile

echo "start execution of /etc/rc.startsap_prod_db" >> $OUT
echo `date` >> $OUT

#-----------------------------------------------------------------------
# Remove old DB-Log-Files
# Remove old Listener-Log-Files (used starting with version 3.0D)
# Remove old Collector's and old Sender's logfiles
# Start the ORACLE Listener (used starting with version 3.0D)
# Start Database and R/3 (central)
#-----------------------------------------------------------------------
if [ -f /oracle/$SAPSID/dbs/sgadef$SAPSID.* ]
then
   /usr/bin/rm /oracle/$SAPSID/dbs/sgadef$SAPSID.*
fi
```

```
if [ -d /tmp/.oracle ]
then
  /usr/bin/rm -r /tmp/.oracle
  /usr/bin/rm /tmp/o
fi

if [ -f /usr/sap/$SAPSID/DVEBMGS$SAPNR/data/rslg* ]
then
   /usr/bin/rm /usr/sap/$SAPSID/DVEBMGS$SAPNR/data/rslg*
fi

/usr/bin/su - $ORAADM "-c /oracle/$SAPSID/bin/lsnrctl start" >> $OUT
# /usr/bin/su - $SAPADM "-c /home/$SAPADM/startsap_${DBSERVER}_$SAPNR ALL"
>> $OUT
/usr/bin/su - $SAPADM "-c /sapmnt/$SAPSID/exe/startdb " >> OU

exit 0
```

A.1.2 Stop script: rc.stopsap_prod_db

The following script stops the database:

```
#!/bin/ksh
#-----------------------------------------------------------------------
# Filename:rc.stopsap_prod_db
# Path:/etc
# Node(s):HACMP-Server and Database-Server
# Info:Stops the Database, R/3 and the 'orasrv' or the 'listener'
#-----------------------------------------------------------------------
# Remark:
# =======
# This script is prepared for SAP R/3 Version 3.0x
#
#-----------------------------------------------------------------------
# First set the environment and dump a timestamp
#-----------------------------------------------------------------------

. /usr/sbin/cluster/saptools/hacmpr3.profile

echo "start execution of /etc/rc.stopsap_prod_db" >> $OUT
echo `date` >> $OUT

#-----------------------------------------------------------------------
# Stop Database and R/3
# Stop orasrv (up to Version 3.0C) or listener (Version 3.0D or higher)
#  - remove the line not needed !!
# Stop saposcol (if running)
```

```
#-----------------------------------------------------------------------
# /usr/bin/su - $SAPADM "-c /home/$SAPADM/stopsap_${DBSERVER}_$SAPNR ALL"
>> $OUT
/usr/bin/su - $SAPADM "-c /sapmnt/$SAPSID/exe/stopdb" >> $OUT
/usr/bin/su - $ORAADM "-c /oracle/$SAPSID/bin/lsnrctl stop" >> $OUT
/usr/bin/su - $SAPADM "-c /oracle/$SAPSID/bin/tcpctl stop" >> $OUT

ps -ef | grep saposcol | fgrep -v grep > /dev/null 2>&1
if [ $? = 0 ]
then
   /usr/bin/su - root "-c /sapmnt/$SAPSID/exe/saposcol -k > /dev/null 2>&1"
fi

exit 0
```

A.2 Starting the CI server

The following scripts are used to start the central instance on the CI server, which is a backup of the DB server.

A.2.1 Start script: rc.startsap_apl

The following script starts the R/3:

```
#!/bin/ksh
#-----------------------------------------------------------------------
# Filename:rc.startsap_apl
# Path:/etc
# Node(s):Applicationserver
# Info:Starts R/3
#-----------------------------------------------------------------------
# Remark:
# =======
# This script is prepared for SAP R/3 Version 3.0x
#
#-----------------------------------------------------------------------
# First set the environment and dump a timestamp
#-----------------------------------------------------------------------

. /usr/sbin/cluster/saptools/hacmpr3.profile

SAPNR=`ls /usr/sap/$SAPSID | grep "D" | fgrep -v "DVEBMGS" | cut -c2-`

echo "start execution of /etc/rc.startsap_apl" >> $OUT
echo `date` >> $OUT
```

```
#-----------------------------------------------------------------------
# If SAP not yet startet, start now
#-----------------------------------------------------------------------
ps -ef | grep dw.sap$SAPSID | fgrep -v grep >/dev/null 2>&1
if [ $? = 0 ]
then
   echo "R/3 is already running" >> $OUT
else

#-----------------------------------------------------------------------
--
# If NFS-Filesystems are not mounted,
# test NFS-Fileserver and mount SAP-Filesystems
#-----------------------------------------------------------------------
if [ ! -f /sapmnt/$SAPSID/profile/DEFAULT.PFL ]
then
   $RSH $NFSSERVER "ps -ef" | grep mountd >/dev/null 2>&1
   if [ $? != 0 ]
   then
      echo "NFS Server $NFSSERVER not available" >> $OUT
      exit 1
   fi

   /usr/sbin/mount -t sap_nfs
   if [ $? != 0 ]
   then
      echo "Failure during mount of SAP related NFS-Filesystems" >> $OUT
      exit 1
   fi
fi

#-----------------------------------------------------------------------
# Check Message-Server at the central R/3 Server
#-----------------------------------------------------------------------
$RSH $MSGSERVER_IP "ps -ef" | grep ms.sap >/dev/null 2>&1
if [ $? != 0 ]
then
   echo "Message Server $MSGSERVER not available" >> $OUT
   exit 1
fi

#-----------------------------------------------------------------------
# Start R/3
#-----------------------------------------------------------------------
/usr/bin/su - $SAPADM "-c /home/$SAPADM/startsap_`hostname`_$SAPNR R3" >>
$OUT
```

```
#-------------------------------------------------------------------------
# If SAP not startet -> ready
#-------------------------------------------------------------------------
fi

#-------------------------------------------------------------------------
# ready -> exit
#-------------------------------------------------------------------------
exit 0
```

A.2.2 Stop script: rc.stopsap_apl

The following script stops R/3 and saposcol:

```
#!/bin/ksh
#-------------------------------------------------------------------------
# Filename:rc.stopsap_apl
# Path:/etc
# Node(s):Applicationserver
# Info:Stops R/3 and saposcol; forced if necessary
#-------------------------------------------------------------------------
# Remark:
# =======
# This script is prepared for SAP R/3 Version 3.0x
#
#-------------------------------------------------------------------------
# First set the environment and dump a timestamp
#-------------------------------------------------------------------------

. /usr/sbin/cluster/saptools/hacmpr3.profile

SAPNR=`ls /usr/sap/$SAPSID | grep "D" | fgrep -v "DVEBMGS" | cut -c2-`
SAPINST=D$SAPNR

echo "start execution of /etc/rc.stopsap_apl" >> $OUT
echo `date` >> $OUT

#-------------------------------------------------------------------------
# If NFS-Server down, don't do anything
#-------------------------------------------------------------------------
if [ -f /sapmnt/$SAPSID/profile/DEFAULT.PFL ]
then

#-------------------------------------------------------------------------
# Stop R3
# Stop saposcol, if running
#-------------------------------------------------------------------------
```

```
/usr/bin/su - $SAPADM "-c /home/$SAPADM/stopsap_`hostname`_$SAPNR R3" >>
$OUT

ps -ef | grep saposcol | fgrep -v grep > /dev/null 2>&1
if [ $? = 0 ]
then
   /usr/bin/su - root "-c /sapmnt/$SAPSID/exe/saposcol -k > /dev/null 2>&1"
fi

#-----------------------------------------------------------------------
# Stop all R3 Processes left running
# Clean shared memory
# Remove Logfiles
#-----------------------------------------------------------------------
ps -ef | grep $SAPINST | fgrep -v grep > /dev/null 2>&1
if [ $? = 0 ];then
   SAPDWPID=`ps -ef | grep dw.sa[p]"$SAPSID"_$SAPINST | awk '{ print $2 }'`
   for i in $SAPDWPID
   do
      kill -2 $i
   done
   SAPSEPID=`ps -ef | grep se.sa[p]"$SAPSID"_$SAPINST | awk '{ print $2 }'`
   for i in $SAPSEPID
   do
      kill -2 $i
   done
   SAPCOPID=`ps -ef | grep co.sa[p]"$SAPSID"_$SAPINST | awk '{ print $2 }'`
   for i in $SACOSPID
   do
      kill -2 $i
   done
   SAPMSPID=`ps -ef | grep ms.sa[p]"$SAPSID"_$SAPINST | awk '{ print $2 }'`
   for i in $SAPMSPID
   do
      kill -2 $i
   done
   SAPPID=`ps -ef | grep START_$SAPINST | fgrep -v grep | awk '{ print $2
}'`
   for i in $SAPPID
   do
      kill -2 $i
   done
   SAPID=`ps -ef | grep "$SAPSID"_$SAPINST | fgrep -v grep | awk '{ print $2
}'`
   for i in $SAPID
   do
      kill -2 $i
   done
fi

#----------------------------------------------------------------------
# Clear shared semaphores/memory
#----------------------------------------------------------------------
/usr/bin/su - $SAPADM "-c /sapmnt/$SAPSID/exe/cleanipc $SAPNR remove" >>
$OUT

#----------------------------------------------------------------------
# If existing, remove Collector's logfiles
#----------------------------------------------------------------------
if [ -f /usr/sap/$SAPSID/$SAPINST/data/rslg* ]
then
   /usr/bin/rm /usr/sap/$SAPSID/$SAPINST/data/rslg*
fi

#----------------------------------------------------------------------
# If NFS-Server down -> ready
#----------------------------------------------------------------------
fi

#----------------------------------------------------------------------
# ready -> exit
#----------------------------------------------------------------------
exit 0
```

A.3 Scripts for SAP application servers

The following scripts are used for the SAP application servers.

A.3.1 Start script: start_clients

```
#!/bin/ksh
#----------------------------------------------------------------------
# Filename:start_clients
# Path:/usr/sbin/cluster/saptools
# Node(s):HACMP-Server; Production and Takeover
# Info:Startscript for the HACMP-Server
#----------------------------------------------------------------------
# First set the environment and dump a timestamp
#----------------------------------------------------------------------

. /usr/sbin/cluster/saptools/hacmpr3.profile

echo "start execution of /usr/sbin/cluster/saptools/start_clients" >> $OUT
echo `date` >> $OUT

#-------------------------------------------------------------------------
# - export all NFS-Filesystems with the right Permissions (/etc/exports)
# - if Takeover happend:
#   - stop all Application-Server
#   - stop own Database
#   - copy the correct Listener-Profile (used starting with version 3.0D)
# - start the Production Database and R/3
# - start all Application-Server
#-------------------------------------------------------------------------
/usr/sbin/exportfs -a

if [ `hostname` = $TAKEOVER ]
then
   /usr/sbin/cluster/saptools/stopsap_apl >> $OUT
   /etc/rc.stopsap_stdby_db >> $OUT
   /usr/bin/cp /usr/sap/trans/tnsnames.ora_$SAPSID
/usr/sap/trans/tnsnames.ora
   /usr/bin/cp /etc/listener.ora_$SAPSID /etc/listener.ora
fi

/etc/rc.startsap_prod_db >> $OUT
/usr/sbin/cluster/saptools/startsap_apl >> $OUT

exit 0
```

A.3.2 Stop script: stop_clients

```
#!/bin/ksh
#------------------------------------------------------------------ -------
# Filename:stop_clients
# Path:/usr/sbin/cluster/saptools
# Node(s):HACMP-Server; Production and Takeover
# Info:Stopscript for the HACMP-Server
#-------------------------------------------------------------------------
# First set the environment and dump a timestamp
#-------------------------------------------------------------------------

. /usr/sbin/cluster/saptools/hacmpr3.profile

echo "start execution of /usr/sbin/cluster/saptools/stop_clients" >> $OUT
echo `date` >> $OUT

#-------------------------------------------------------------------------
# - stop all Application-Server
# - stop the Production Database and R/3
# - if Takeover happend before:
#   - copy the correct Listener-Profile (used starting with version 3.0D)
#   - start the own Database and R/3
#-----------------------------------------------------------------------
/usr/sbin/cluster/saptools/stopsap_apl >> $OUT
/etc/rc.stopsap_prod_db >> $OUT
if [ `hostname` = $TAKEOVER ]
then
   /usr/bin/cp /usr/sap/trans/tnsnames.ora_$STBYSID
/usr/sap/trans/tnsnames.ora
   /usr/bin/cp /etc/listener.ora_$STBYSID /etc/listener.ora
   /etc/rc.startsap_stdby_db >> $OUT
fi

exit 0
```

A.4 Scripts for client processes

The following scripts are used for client processes.

A.4.1 Start script for DB server: start_db_clients

```
#!/bin/ksh
#-----------------------------------------------------------------------
# Filename:start_db_clients
# Path:/usr/sbin/cluster/saptools
# Node(s):HACMP-Server and DB-Server; Production and Takeover
# Info:Startscript for the DB-Server
#-----------------------------------------------------------------------
# First set the environment and dump a timestamp
#-----------------------------------------------------------------------

. /usr/sbin/cluster/saptools/hacmpr3.profile

echo "start execution of /usr/sbin/cluster/saptools/start_db_clients" >>
$OUT
echo `date` >> $OUT

#-----------------------------------------------------------------------
# - export all NFS-Filesystems with the right Permissions (/etc/exports)
# - if Takeover happend:
#   - stop all Application-Server
#   - stop own R/3
# - start the Database
# - start the Central-Server (which starts all Application-Server)
#-----------------------------------------------------------------------
```

```
/usr/sbin/exportfs -a

if [ `hostname` = $MSGSERVER ]
then
   /usr/sbin/cluster/saptools/stopsap_apl >> $OUT
   /etc/rc.stopsap_ci >> $OUT
   /etc/rc.startsap_dbci >> $OUT
   /usr/sbin/cluster/saptools/startsap_apl >> $OUT
else
   /etc/rc.startsap_db >> $OUT
   DOING="echo \"/usr/sbin/cluster/saptools/start_r3_clients >> $OUT\" | at
now"
   $RSH $MSGSERVER_IP $DOING
fi

exit 0
```

A.4.2 Stop script for DB server: stop_db_clients

```
#!/bin/ksh
#-------------------------------------------------------------------------
# Filename:stop_db_clients
# Path:/usr/sbin/cluster/saptools
# Node(s):HACMP-Server and DB-Server; Production and Takeover
# Info:Stopscript for the DB-Server
#-------------------------------------------------------------------------
# First set the environment and dump a timestamp
#-------------------------------------------------------------------------

. /usr/sbin/cluster/saptools/hacmpr3.profile

echo "start execution of /usr/sbin/cluster/saptools/stop_db_clients" >>
$OUT
echo `date` >> $OUT

#-------------------------------------------------------------------------
# - stop the Central-Server (which stops all Application-Server)
# - stop the Database
#-------------------------------------------------------------------------
if [ `hostname` = $MSGSERVER ]
then
   /usr/sbin/cluster/saptools/stopsap_apl >> $OUT
else
   DOING="echo \"/usr/sbin/cluster/saptools/stop_r3_clients >> $OUT\" | at
now"
   $RSH $MSGSERVER_IP $DOING
fi
```

```
/etc/rc.stopsap_db >> $OUT

exit 0
```

A.4.3 Start script for CI server: start_r3_clients

```
#!/bin/ksh
#------------------------------------------------------------------------
# Filename:start_r3_clients
# Path:/usr/sbin/cluster/saptools
# Node(s):HACMP-Server and DB-Server; Production and Takeover
# Info:Startscript for the R/3 Central-Server
#------------------------------------------------------------------------
# First set the environment and dump a timestamp
#------------------------------------------------------------------------

. /usr/sbin/cluster/saptools/hacmpr3.profile

echo "start execution of /usr/sbin/cluster/saptools/start_r3_clients" >>
$OUT
echo `date` >> $OUT

#------------------------------------------------------------------------
# - if Takeover happend:
#   - stop all Application-Server
#   - wait 15 seconds until the Application-Server has stopped R/3
# - start Central-Server
# - start all Application-Server
#------------------------------------------------------------------------

if [ `hostname` = $DBSERVER ]
then
   /usr/sbin/cluster/saptools/stopsap_apl >> $OUT
   /usr/bin/sleep 15
fi

/etc/rc.startsap_ci >> $OUT
/usr/sbin/cluster/saptools/startsap_apl >> $OUT

exit 0
```

A.4.4 Stop script for CI server: stop_r3_clients

```
#!/bin/ksh
#------------------------------------------------------------------------
# Filename:stop_r3_clients
# Path:/usr/sbin/cluster/saptools
# Node(s):HACMP-Server and DB-Server; Production and Takeover
# Info:Stopscript for the R/3 Central-Server
#-----------------------------------------------------------------------
--
# First set the environment and dump a timestamp
#-----------------------------------------------------------------------
--

. /usr/sbin/cluster/saptools/hacmpr3.profile

echo "start execution of /usr/sbin/cluster/saptools/stop_r3_clients" >>
$OUT
echo `date` >> $OUT

#-----------------------------------------------------------------------
# - stop all Application-Server
# - stop R/3 at Central-Server
#-----------------------------------------------------------------------
/usr/sbin/cluster/saptools/stopsap_apl >> $OUT
/etc/rc.stopsap_ci >> $OUT

exit 0
```

A.5 Scripts for central instance

The following scripts are used for the central instance.

A.5.1 Start scripts

```
#!/bin/ksh
#-----------------------------------------------------------------------
# Filename:rc.startsap_ci
# Path:/etc
# Node(s):Central-Server
# Info:Starts R/3
#-----------------------------------------------------------------------
# Remark:
# =======
# This script is prepared for SAP R/3 Version 3.0x
#
#-----------------------------------------------------------------------
# First set the environment and dump a timestamp
#-----------------------------------------------------------------------

. /usr/sbin/cluster/saptools/hacmpr3.profile
```

```
SAPINST=DVEBMGS$SAPNR

echo "start execution of /etc/rc.startsap_ci" >> $OUT
echo `date` >> $OUT

#--------------------------------------------------------------------------
# If HACMP not yet startet -> exit
#--------------------------------------------------------------------------
ps -ef | grep clstrmgr | fgrep -v grep >/dev/null 2>&1
if [ $? != 0 ]
then
   echo "HACMP not running at localhost" >> $OUT
   exit 1
else

#--------------------------------------------------------------------------
# If SAP not yet startet, start now
#--------------------------------------------------------------------------
ps -ef | grep dw.sap$SAPSID | fgrep -v grep >/dev/null 2>&1
if [ $? = 0 ]
then
   echo "R/3 is already running" >> $OUT
else

#--------------------------------------------------------------------------
# If NFS-Filesystems are not mounted,
# test NFS-Fileserver and mount SAP-Filesystems
#--------------------------------------------------------------------------
if [ ! -f /sapmnt/$SAPSID/profile/DEFAULT.PFL ]
then
   $RSH $NFSSERVER "ps -ef" | grep mountd >/dev/null 2>&1
   if [ $? != 0 ]
   then
      echo "NFS Server $NFSSERVER not available" >> $OUT
      exit 1
   fi

   /usr/sbin/mount -t sap_nfs
   if [ $? != 0 ]
   then
      echo "Failure during mount of $i" >> $OUT
      exit 1
   fi
fi

#--------------------------------------------------------------------------
```

```
# If ORACLE-server not running -> exit
# Test the listener (3.0D or higher) or the orasrv (up to 3.0C)
#  - remove the line not needed !
#----------------------------------------------------------------------
$RSH $DBSERVER_IP "ps -ef" | grep orasrv >/dev/null 2>&1
$RSH $DBSERVER_IP "ps -ef" | grep tnslsnr >/dev/null 2>&1
if [ $? != 0 ]
then
   echo "ORACLE Server $DBSERVER not running" >> $OUT
   exit 1
else

#----------------------------------------------------------------------
# If existing, remove Collector's logfiles
# Start R/3
#----------------------------------------------------------------------
if [ -f /usr/sap/$SAPSID/$SAPINST/data/rslg* ]
then
   /usr/bin/rm /usr/sap/$SAPSID/$SAPINST/data/rslg*
fi

/usr/bin/su - $SAPADM "-c /home/$SAPADM/startsap_`hostname`_$SAPNR R3" >>
$OUT

#----------------------------------------------------------------------
# If orasrv not running -> ready
#----------------------------------------------------------------------
fi

#----------------------------------------------------------------------
# If SAP not startet -> ready
#----------------------------------------------------------------------
fi

#----------------------------------------------------------------------
# If HACMP not startet -> ready
#----------------------------------------------------------------------
fi

#----------------------------------------------------------------------
# ready -> exit
#----------------------------------------------------------------------
exit 0
```

A.5.2 Stop scripts

```
#!/bin/ksh
```

```
#----------------------------------------------------------------------------
# Filename:rc.stopsap_ci
# Path:/etc
# Node(s):Central-Server
# Info:Stops R/3 and saposcol; forced if necessary
#----------------------------------------------------------------------------
# Remark:
# =======
# This script is prepared for SAP R/3 Version 3.0x
#
#----------------------------------------------------------------------------
# First set the environment and dump a timestamp
#----------------------------------------------------------------------------

. /usr/sbin/cluster/saptools/hacmpr3.profile

SAPINST=DVEBMGS$SAPNR

echo "start execution of /etc/rc.stopsap_ci" >> $OUT
echo `date` >> $OUT

#----------------------------------------------------------------------------
# If NFS-Server down, don't do anything
#----------------------------------------------------------------------------
if [ -f /sapmnt/$SAPSID/profile/DEFAULT.PFL ]
then

#----------------------------------------------------------------------------
# Stop R3
# Stop saposcol, if running - but only at the original R/3 central Server
#----------------------------------------------------------------------------
/usr/bin/su - $SAPADM "-c /home/$SAPADM/stopsap_`hostname`_$SAPNR R3" >>
$OUT

if [ `hostname` = $MSGSERVER ]
then
   ps -ef | grep saposcol | fgrep -v grep > /dev/null 2>&1
   if [ $? = 0 ]
   then
      /usr/bin/su - root "-c /sapmnt/$SAPSID/exe/saposcol -k > /dev/null
2>&1"
   fi
fi

#----------------------------------------------------------------------------
# Stop all R3 Processes left running
# Clean shared memory
# Remove Logfiles
#----------------------------------------------------------------------
ps -ef | grep $SAPINST | fgrep -v grep > /dev/null 2>&1
if [ $? = 0 ];then
   SAPDWPID=`ps -ef | grep dw.sa[p]"$SAPSID"_$SAPINST | awk '{ print $2 }'`
   for i in $SAPDWPID
   do
      kill -2 $i
   done
   SAPSEPID=`ps -ef | grep se.sa[p]"$SAPSID"_$SAPINST | awk '{ print $2 }'`
   for i in $SAPSEPID
   do
      kill -2 $i
   done
   SAPCOPID=`ps -ef | grep co.sa[p]"$SAPSID"_$SAPINST | awk '{ print $2 }'`
   for i in $SACOSPID
   do
      kill -2 $i
   done
   SAPMSPID=`ps -ef | grep ms.sa[p]"$SAPSID"_$SAPINST | awk '{ print $2 }'`
   for i in $SAPMSPID
   do
      kill -2 $i
   done
   SAPPID=`ps -ef | grep START_$SAPINST | fgrep -v grep | awk '{ print $2
}'`
   for i in $SAPPID
   do
      kill -2 $i
   done
   SAPID=`ps -ef | grep "$SAPSID"_$SAPINST | fgrep -v grep | awk '{ print $2
}'`
   for i in $SAPID
   do
      kill -2 $i
   done
fi

#----------------------------------------------------------------------
# Clear shared semaphores/memory
#----------------------------------------------------------------------
/usr/bin/su - $SAPADM "-c /sapmnt/$SAPSID/exe/cleanipc $SAPNR remove" >>
$OUT

#----------------------------------------------------------------------
# If existing, remove Collector's logfiles
#----------------------------------------------------------------------
```

```
if [ -f /usr/sap/$SAPSID/$SAPINST/data/rslg* ]
then
   /usr/bin/rm /usr/sap/$SAPSID/$SAPINST/data/rslg*
fi

#-----------------------------------------------------------------------
# If NFS-Server down -> ready
#-----------------------------------------------------------------------
fi

#-----------------------------------------------------------------------
# ready -> exit
#-----------------------------------------------------------------------
exit 0
```

A.5.3 Start scripts

```
#!/bin/ksh
#-----------------------------------------------------------------------
# Filename:rc.startsap_db
# Path:/etc
# Node(s):HACMP-Server and Database-Server
# Info:Starts the Database
#-----------------------------------------------------------------------
# Remark:
# =======
# This script is prepared for SAP R/3 Version 3.0x
#
#-----------------------------------------------------------------------
# First set the environment and dump a timestamp
#-----------------------------------------------------------------------

. /usr/sbin/cluster/saptools/hacmpr3.profile

echo "start execution of /etc/rc.startsap_db" >> $OUT
echo `date` >> $OUT

#-----------------------------------------------------------------------
# Remove old DB-Log-Files
# Remove old Listener-Log-Files (used starting with version 3.0D)
# Start the ORACLE Listener (used starting with version 3.0D)
# Start Database
#-----------------------------------------------------------------------
if [ -f /oracle/$SAPSID/dbs/sgadef$SAPSID.* ]
then
   /usr/bin/rm /oracle/$SAPSID/dbs/sgadef$SAPSID.*
fi
```

```
if [ -d /tmp/.oracle ]
then
  /usr/bin/rm -r /tmp/.oracle
fi

/usr/bin/su - $ORAADM "-c /oracle/$SAPSID/bin/lsnrctl start" >> $OUT
/usr/bin/su - $SAPADM "-c /home/$SAPADM/startsap_`hostname`_$SAPNR DB" >>
$OUT

exit 0
```

A.5.4 Stop scripts

```
#!/bin/ksh
#-----------------------------------------------------------------------
# Filename:rc.startsap_db
# Path:/etc
# Node(s):HACMP-Server and Database-Server
# Info:Starts the Database
#-----------------------------------------------------------------------
# Remark:
# =======
# This script is prepared for SAP R/3 Version 3.0x
#
#-----------------------------------------------------------------------
# First set the environment and dump a timestamp
#-----------------------------------------------------------------------

. /usr/sbin/cluster/saptools/hacmpr3.profile

echo "start execution of /etc/rc.startsap_db" >> $OUT
echo `date` >> $OUT

#-----------------------------------------------------------------------
# Remove old DB-Log-Files
# Remove old Listener-Log-Files (used starting with version 3.0D)
# Start the ORACLE Listener (used starting with version 3.0D)
# Start Database
#-----------------------------------------------------------------------
if [ -f /oracle/$SAPSID/dbs/sgadef$SAPSID.* ]
then
   /usr/bin/rm /oracle/$SAPSID/dbs/sgadef$SAPSID.*
fi

if [ -d /tmp/.oracle ]
then
```

```
  /usr/bin/rm -r /tmp/.oracle
fi

/usr/bin/su - $ORAADM "-c /oracle/$SAPSID/bin/lsnrctl start" >> $OUT
/usr/bin/su - $SAPADM "-c /home/$SAPADM/startsap_`hostname`_$SAPNR DB" >>
$OUT

exit 0
```

A.5.5 Start script: rc_startsap_dbci

The following script starts the database and central instance after DB server takeover:

```
#!/bin/ksh
#-------------------------------------------------------------------------
# Filename:rc.startsap_dbci
# Path:/etc
# Node(s):HACMP-Server and Database-Server
# Info:Starts the Database and Central Instance after DB-takeover
#-------------------------------------------------------------------------
# Remark:
# =======
# This script is prepared for SAP R/3 Version 3.0x
#
#-------------------------------------------------------------------------
# First set the environment and dump a timestamp
#-------------------------------------------------------------------------

. /usr/sbin/cluster/saptools/hacmpr3.profile

SAPINST=DVEBMGS$SAPNR

echo "start execution of /etc/rc.startsap_dbci" >> $OUT
echo `date` >> $OUT

#-------------------------------------------------------------------------
# If existing, remove old DB-Log-Files
# If existing, remove old Listener-Log-Files (used starting with version
3.0D)
# If existing, remove Collector's logfiles
# Start the ORACLE Listener (used starting with version 3.0D)
# Start Database and Central Instance using a different instance profile
#-------------------------------------------------------------------------
if [ -f /oracle/$SAPSID/dbs/sgadef$SAPSID.* ]
then
   /usr/bin/rm /oracle/$SAPSID/dbs/sgadef$SAPSID.*
```

```
fi

if [ -d /tmp/.oracle ]
then
  /usr/bin/rm -r /tmp/.oracle
fi

if [ -f /usr/sap/$SAPSID/$SAPINST/data/rslg* ]
then
   /usr/bin/rm /usr/sap/$SAPSID/$SAPINST/data/rslg*
fi

/usr/bin/su - $ORAADM "-c /oracle/$SAPSID/bin/lsnrctl start" >> $OUT
/usr/bin/su - $SAPADM "-c /home/$SAPADM/startsap_${DBSERVER}_$SAPNR ALL" >>
$OUT

exit 0
```

A.6 Scripts on takeover nodes

The following scripts run on the takeover nodes during the failover and reintegration stage.

A.6.1 Start script: rc.startsap_stdby_db

```
#!/bin/ksh
#-------------------------------------------------------------------------
# Filename:rc.startsap_stdby_db
# Path:/etc
# Node(s):HACMP-Server and Database-Server
# Info:Starts the Database and R/3
#-------------------------------------------------------------------------
# Remark:
# =======
# This script is prepared for SAP R/3 Version 3.0x
#
#-------------------------------------------------------------------------
# First set the environment and dump a timestamp
#-------------------------------------------------------------------------

. /usr/sbin/cluster/saptools/hacmpr3.profile

echo "start execution of /etc/rc.startsap_stdby_db" >> $OUT
echo `date` >> $OUT

#-------------------------------------------------------------------------
```

```
# Next line is for HACMP Reasons only (to start DB automacticly):
#    Remove old Listener-Log-Files (used starting with version 3.0D)
# Start the ORACLE Listener (used starting with version 3.0D)
# Start Database and R/3 (central)
#--------------------------------------------------------------------------
if [ -d /tmp/.oracle ]
then
  /usr/bin/rm -r /tmp/.oracle
  /usr/bin/rm /tmp/o
fi

/usr/bin/su - $ORASTBY "-c /oracle/$STBYSID/bin/lsnrctl start" >> $OUT
/usr/bin/su - $STBYADM "-c /home/$STBYADM/startsap_${TAKEOVER}_$STBYNR ALL"
>> $OUT

exit 0
```

A.6.2 Stop script: rc.stopsap_stdby_db

```
#!/bin/ksh
#--------------------------------------------------------------------------
# Filename:rc.stopsap_stdby_db
# Path:/etc
# Node(s):HACMP-Server and Database-Server
# Info:Stops the Database, R/3 and the 'orasrv' or the 'listener'
#--------------------------------------------------------------------------
# Remark:
# =======
# This script is prepared for SAP R/3 Version 3.0x
#
#--------------------------------------------------------------------------
# First set the environment and dump a timestamp
#--------------------------------------------------------------------------

. /usr/sbin/cluster/saptools/hacmpr3.profile

echo "start execution of /etc/rc.stopsap_stdby_db" >> $OUT
echo `date` >> $OUT

#--------------------------------------------------------------------------
# Stop Database and R/3
# Stop orasrv (up to Version 3.0C) or listener (Version 3.0D or higher)
#  - remove the line not needed !
# Stop saposcol (if running)
#--------------------------------------------------------------------------
/usr/bin/su - $STBYADM "-c /home/$STBYADM/stopsap_${TAKEOVER}_$STBYNR ALL"
>> $OUT
```

```
/usr/bin/su - $ORASTBY "-c /oracle/$STBYSID/bin/lsnrctl stop" >> $OUT
/usr/bin/su - $STBYADM "-c /oracle/$STBYSID/bin/tcpctl stop" >> $OUT

ps -ef | grep saposcol | fgrep -v grep > /dev/null 2>&1
if [ $? = 0 ]
then
   /usr/bin/su - root "-c /sapmnt/$STBYSID/exe/saposcol -k > /dev/null
2>&1"
fi

exit 0
```

A.7 Profile script

The following script is called by all the previous scripts to initialize the enviroment variables:

```
#!/bin/ksh
#-------------------------------------------------------------
# Filename:hacmpr3.profile
# Path:/usr/sbin/cluster/saptools
# Node(s):All R/3-Server
# Info:Main profile to setup HACMP and R/3 variables
#-------------------------------------------------------------
# Part 1
# Variables to be setup to meet your HACMP - R/3 installation.
# May be there are some you don't really need !
#--------------------------------------------------------------

# R/3 environment
#================
SAPSID=PRD        # Set SAP SID of productiv R/3
SAPNR=00          # Set SAP Nr of productiv R/3

STBYSID=TST       # Set SAP SID of Takeoversystem
STBYNR=00         # Set SAP Nr of Takeoversystem

# Hostinformations
#=================
DBSERVER=prdsrv   # Set hostname of Database-Server
MSGSERVER=prdsrv  # Set hostname of R/3 Central-Server

TAKEOVER=tstsrv   # Set hostname of Takeover-Host

# IP-Label (=> IP-address)
#=========================
NFSSERVER=prdsrv_srvs     # Set service IP-label of NFS-Server
DBSERVER_IP=prdsrv_srvs   # Set service IP-label of Database-Server
MSGSERVER_IP=prdsrv_srvs  # Set service IP-label of R/3 Central-Server

# Licensekey    # needed for SAP R/3 Version 2.x only
#===========
INSTALL_NR=0047110815
EXPIRATION_DATE=99991231
TWO_TASK=T:$DBSERVER:$SAPSID

# Remote shell command "rsh"
# e.g. choose Kerberos version for SP2
#====================================
RSH=/usr/bin/rsh           # "normal" rsh
#RSH=/usr/lpp/ssp/rcmd/bin/rsh    # Kerberos Version of rsh

#----------------------------------------------------------------
# Part 2
# Standard variables used in the scripts, but not to be changed.
# Don't forget to control the filenames and their locations.
#----------------------------------------------------------------

# R/3 environment
#================
SAPLOWID=`echo $SAPSID | /usr/bin/tr '[:upper:]' '[:lower:]'`
SAPADM="$SAPLOWID"adm
ORAADM=ora$SAPLOWID
DB2ADM=db2$SAPLOWID

STBYLOWID=`echo $STBYSID | /usr/bin/tr '[:upper:]' '[:lower:]'`
STBYADM="$STBYLOWID"adm
ORASTBY=ora$STBYLOWID
DB2STBY=db2$STBYLOWID

# DB2 environment
#================
DB2AS=db2as

# Logfile
#========
OUT=/tmp/hacmp.out
```

Appendix B. Domino HACMP script and programs

The two Domino application server scripts, call_start_srv1 and call_stop_srv1, listed here are for Domino server instance 1. For server instance 2, replace all index 1 with index 2. The source code hadomtest.c for the Domino test program hadomtest is also listed in this appendix.

B.1 call_start_srv1 script

```
#!/bin/ksh
#
#Shell script to call the start Domino server in a HACMP environment
#02022000 IBM ITSO Austin,   Ole Conradsen
#
# this script calls the start script as the user notes1 to
# start the notes server
su - notes1 -c /home/notes1/start_server1
nohup /u/notes1/check_server  &
exit 0
```

B.2 call_stop_srv1 script

```
#!/bin/ksh
#
#Shell script to call the stop Domino server in a HACMP environment
#02022000 IBM ITSO Austin,   Ole Conradsen
#
# this script calls the stop script as the user notes1
# we don't want to run the stop script as root in a HACMP env
# it kills processes and shared memory
#
#First kill the monitor process, and then stop notes

ps -ef | grep check_server |  awk '{ print $2 }' | xargs kill -9
sleep 2
su - notes1 -c /home/notes1/stop_server1
exit 0
```

B.3 start_server1 script

Start_server1 is the shell script used to start partition one of the Domino server. Before we start the server ,we check that the script is actually running as user notes1. The reason for this is that the file permissions will be altered in the data directory if we start the server as another user, and then may possibly prevent it from running as user notes1. The output is sent to a file, notesserver.log, but the information in the log file can also be retrieved from the Domino log.nsf database. The log file, therefore, can be redirected to /dev/null without losing information.

```
#!/bin/ksh
#
#Shell script to start Domino server in a HACMP environment
#02022000 IBM ITSO Austin
#
echo  $LOGNAME | grep notes1 > /dev/null
if [ $? != 0 ]
then
echo "not a accepted Domino user ! "
echo ""
exit 1
fi
mv /server1/notesserver.log.1 /server1/notesserver.log.1 2>/dev/null
mv /server1/notesserver.log /server1/notesserver.log.1 2>/dev/null
nohup /opt/lotus/bin/server   > /server1/notesserver.log 2>&1 &
exit 0
```

B.4 stop_server1 script

Stop_server1 is the shell script used to stop the Domino server. Before we stop the server, we check that the script is actually running as user notes1. The reason for this is security. The script will search for semaphores and shared memory segments and remove those belonging to user notes1. If something should go wrong, the system will prevent us from removing other resources if we are running as notes1 and not as root.

```
#!/bin/ksh
#
#Shell script to stop Domino server in a HACMP environment
#02022000 IBM ITSO Austin, OLC
#
PATH=/server1:$PATH
export PATH
echo  $LOGNAME | grep notes1 > /dev/null
if [ $? != 0 ]
then
echo "not a accepted Domino user ! "
exit 1
fi
/opt/lotus/bin/server  -q &
```

```
sleep 120
for i in `ps -ef | grep notes1 |grep -v ksh | grep -v grep | grep -v 'ps
-ef' | \
awk '{ print $2 }' `
do
#       echo "removing process  " $i
awk '{ print $2 }' `
do
#        echo "removing process  " $i
kill -9 $i
done
for i in `ipcs | grep notes1 | awk '/^m / {print $2}'`
do
#       echo "removing Shared Memory " $i
ipcrm -m $i done

for i in `ipcs | grep notes1 | awk '/^s / {print $2}'`
do
#        echo "removing Semaphore " $i
ipcrm -s $i
done
```

B.5 check_server script

The check_server program is designed to loop continuously and check that the Domino server can be accessed. The script will work on both bmw and vw as it checks which file systems are mounted locally.

Note

It is very important to export PATH, NOTES_DATA_DIR, LOTUS, and Notes_ExecDirectory in the script or the program /u/notes1/hadomtest won't run.

```
#!/bin/ksh
#
#Shell script to check Domino server in a HACMP environment
#02022000 IBM ITSO Austin
#
# The script is made for a two server configuration bmw and vw, on each
# server this script is running, the script check if the notes data
directory is
# is present for a server, and if it is, we assume that the server should
also
# be running. Some waits are added to let the Domino server start, after
# checking the Domino data, we wait 120 seconds to make sure that the
# Domino
# had time to start. If the server doesn't respond, we stop it and restart
it.
# wait 120 seconds after we find the Domino data directory before polling
# to let the Domino servers start. Also after each look wait 180 seconds
# between poll loops to avoid performance impact from the test program.

export PATH=/opt/lotus/bin:/usr/bin:/etc:/usr/sbin:/usr/ucb: \
$HOME/bin:/usr/bin/X11:/sbin:.:/server1:/server2:

export NOTES_DATA_DIR=/server1
export LOTUS=/opt/lotus
export Notes_ExecDirectory=/usr/domino/lotus/notes/latest/ibmpow

sleep 5
while [ 1 ]
do
if [ -f /server1/notes.ini ]
then
#            /server1 exist and is mounted so check it !!
sleep 120
/u/notes1/hadomtest bmw names.nsf
if [ $? != 0 ]
then
su - notes1 -c  /u/notes1/stop_server1
sleep 2
su - notes1 -c  /u/notes1/start_server1
else
echo " OK DOMINO IS RUNNING ON BMW "
echo " OK DOMINO IS RUNNING ON BMW "
fi
fi
if [ -f /server2/notes.ini ]
then
#            /server2 exist and is mounted so check it !!
sleep 120
/u/notes2/hadomtest vw names.nsf
if [ $? != 0 ]
then
su - notes2 -c /u/notes2/stop_server2
sleep 2
su - notes2 -c /u/notes2/start_server2
else
echo " OK DOMINO IS RUNNING ON VW  "
fi
fi
```

```
sleep 180
done
```

B.6 hadomtest.c source

hadomtest.c is a sample C program written to test the connection to a Domino server. The program is called with two mandatory and one optional parameter <server name> <database filename> [debug level]. The server name is the TCP/IP name and the server name of the Domino server we want to check. The database filename is an arbitrary filename for a Domino database we know exists, for example, names.nsf. Remember, AIX is case sensitive on filenames. The third optional parameter is a number to control the output from the program. Without this parameter, the program does not print anything, it only returns a response code 0, or not 0, for successful and unsuccessful connections, respectively.

To compile the C program, you need the Domino Release 5.0.2. C API and an AIX C compiler.

To execute the program, you need to have the Domino library installed on your system and some and the LIB environment variables pointing to where the library is installed. The Domino library file name on AIX is libnotes_r.a and must be in the lib search path for the user. The user executing the program must also have the following environment variables set:

- PATH, the PATH environment, must include the Domino data directory and the Domino program directory.
- NOTES_DATA_DIR must point to the Domino data directory, in this example, /server1.
- LOTUS must point to the Domino binary root directory, in our example, and in most installations, this is the symbolic link from /opt/lotus made by the Domino installation.
- Notes_ExecDirectory must point to the directory containing the Domino executable, in our example, this is /usr/lpp/lotus/notes/latest/ibmpow.

After setting these environment variables, the user will be able to test the connection to the Domino server by executing the program:

```
/*************************************************************************
PROGRAM:            hadomtest
FILE:               hadomtest.c
PURPOSE:     test the availability of Domino server
                                  02092000 ITSO
*************************************************************************/
```

```
/* OS and C include files */

#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/err_rec.h>
#include <sys/err_rec.h>

/* Notes API include files */
#include "global.h"
#include "nsfdb.h"
#include "nsfdata.h"
#include "osmisc.h"

int     debug;

/* define exit codes if they are not defined elsewhere */

#ifndef EXIT_SUCCESS
   #define EXIT_SUCCESS 0
#endif
#ifndef EXIT_FAILURE
   #define EXIT_FAILURE 1
#endif

char *ctime();

/* Local function prototypes */
void print_api_error (STATUS);
/* global variables */
        time_t seconds,*nu_of_sec;
        char   *tbufp,tbuf[30];

void gettime()
{
nu_of_sec = &seconds;
seconds = time(&seconds);
tbufp = ctime(nu_of_sec);
}

VOID main (int argc, char *argv[])
{
    /* Local data declarations */
```

```
char       filename[30];
   char      fullpath_name[30];           /* pathname of database */
   DBHANDLE   db_handle;             /* database handle */
   char      buffer[NSF_INFO_SIZE];  /* database info buffer */
   char      title[NSF_INFO_SIZE];   /* database title */
   STATUS     error;                  /* error code from API calls */
       int            fd;

/* Get the pathname of the database from the command line. */
if (argc < 3)
    {
        printf( "\nUsage:  %s <server name> <database filename> [ debug
level]\n
", argv[0] );
      exit (EXIT_SUCCESS);
}
debug = 0;
       if (argc == 4 ) debug = atoi(argv[3]);
       if ( debug > 4 ) printf ("debug value = %d\n",debug);

       if (debug > 4)
      printf(" calls  NotesInitExtended  %d params = %s \n",argc,argv[1]);
       error = NotesInitExtended (argc, argv);
       if ( debug > 1 ) printf("error = %d \n",error);
       if (error) {
               if ( debug > 1 )  fprintf (stderr, "\nError initializing
Notes.\
n");
              exit (EXIT_FAILURE);
               }
/* Open the database. */
if (error = OSPathNetConstruct(NULL,argv[1],argv[2],fullpath_name))
       exit(-1);
if ( debug > 4 ) printf(" now we have Path construct  \n");
if ( debug > 4 ) printf(" now we have Path construct  \n");
if (error = NSFDbOpen (fullpath_name, &db_handle)) {
               print_api_error (error);
               NotesTerm();
               exit (EXIT_FAILURE);
               }
/* Get the database title. */
if (error = NSFDbInfoGet (db_handle, buffer)) {
               print_api_error (error);
               NSFDbClose (db_handle);
               NotesTerm();
               exit (EXIT_FAILURE);
        }
```

```
NSFDbInfoParse (buffer, INFOPARSE_TITLE, title, NSF_INFO_SIZE - 1);
/* Print the title. */

if ( debug > 0 )
      printf ("\nThe title of this database is: %s\n\n", title);

       /* Close the database. */
if (error = NSFDbClose (db_handle)) {
              print_api_error (error);
              NotesTerm();
              exit (EXIT_FAILURE);
       }

      /* Terminate Notes. session */

NotesTerm();

* End of main program. */

   exit (EXIT_SUCCESS);



* This function prints the API error message associated with an error code.
*/
void print_api_error (STATUS api_error)

{
    STATUS  string_id = ERR(api_error);
    char    error_text[200];
    WORD    text_len;

    /* Get the message for this API error code from the resource string
       table. */

    text_len = OSLoadString (NULLHANDLE,
                             string_id,
                             error_text,
                             sizeof(error_text));


        if ( debug > 1 ) printf ("\n%s\n", error_text);

}
```

Appendix C. Cluster aware application source codes

In this appendix, we provide the source code of the sample client-server application and cluster sense program discussed in Chapter 15, “Writing a cluster aware application” on page 237.

C.1 Socket send (TX client application)

The sample client program sends a string to a server program and receives back the number of bytes written out to a data file by the server. You must specify the server host name and the string to send as a command line parameter.

Usage: `txclient server_name string`

```
/*
txclinet.c
This program sends a string to the server. The server stores it in a file
and sends back the number of bytes written out to the file.

Usage: txclient server_name string
server_name: Host name or IP address of the server
string: A string to send to the server.
Example: client ford “hello world”
*/

#include <stdio.h>
#include <errno.h>
#include <sys/socket.h>
#include <netdb.h>

/*      TCP/IP port number    */

#define PORT 24259


int call_socket( char * hostname, int portnum ){

/*
        Open socket connection to the server
        Parameters: hostname: server host name
                    portnum: TCP/IP portnumber
        Return value: connection handler
*/
```

```
        struct sockaddr_in sa;
        struct hostent     *hp;
        int s;

        if ((hp=gethostbyname(hostname)) == NULL) {
                errno = ECONNREFUSED;
                return -1;
                }
        bzero(&sa,sizeof(sa));
        bcopy(hp->h_addr,(char *)&sa.sin_addr,hp->h_length);
        sa.sin_family = hp->h_addrtype;
        sa.sin_port = htons((u_short)portnum);
        if ((s= socket(hp->h_addrtype,SOCK_STREAM,0)) < 0) {
                return -1;
                }
        if (connect(s,(struct sockaddr *)&sa,sizeof sa) < 0) {
                close(s);
                return -1;
                }
        return s;
        }


int main( char argc,  char * argv[] ){
        int s;
        char string[255];
        int  bytes;

/*      Command line argument check    */

        if( argc!=3) {
                printf("Usage: %s hostname string\n", argv[0]);
                exit( 1 );
                }

/*      Open connection to the server    */

        s = call_socket((unsigned char *)argv[1], PORT);
        if(s<0){
                printf("Bad server name or server is not running!\n");
                exit(1);
                }
        printf("Connected to server: %s, using port %i\n", argv[1],PORT);

/*      Send the string to the server    */

        strcpy(string,argv[2]);
        printf("The string: %s\n",string);
        write(s,string,strlen(string));

/*      Read the return value (number of bytes written to the data file)
        from the server    */

        read(s,&bytes,sizeof(bytes));
        if(bytes<0){

/*      If the return value is less then 0, then I/O error occurred on the
        server */

                fprintf(stderr,"File I/O error on the server!\n");
                close(s);
                exit(0);
                }

/*      Print out the return value    */

       printf("%d bytes was successfully written to the strings.data file\
on the server\n",bytes);

/*      Close socket connection    */

        close( s );
        exit( 0 );
        }
```

C.2 Socket receive (TX server application)

The following program is our server application. It runs in the background and is capable of multiple client connections. The client program sends a string to the server. The server program stores it in the /sharedfs/app/strings.data file and sends back the number of bytes written out to the client.

Usage: `txserver &`

```
/*
txserver.c
Sample TCP server. It receives a string form its client, stores it in a file
and sends back the number of bytes written in the file
*/

#include <stdio.h>
#include <errno.h>
#include <netinet/in.h>
```

```
#include <sys/types.h>
#include <sys/param.h>
#include <sys/socket.h>
#include <netdb.h>
#include <sys/wait.h>

/*      TCP/IP port number    */

#define PORT 24259

int establish(u_short portnum){
/*
        This function establish the socket connection for listening for
        the clients.
        Parameters: portnum: TCP port number
        Return value: TCP connection handler
*/
        char   myname[MAXHOSTNAMELEN+1];
        int    s;
        struct sockaddr_in sa;
        struct hostent *hp;

        bzero(&sa,sizeof(struct sockaddr_in));
        gethostname(myname, MAXHOSTNAMELEN);

/*      Get the server's host name    */

        hp=gethostbyname(myname);
        if(hp == NULL) return -1;
        sa.sin_family=hp->h_addrtype;
        sa.sin_port=htons(portnum);

/*      Open socket connection    */

        if((s=socket(AF_INET, SOCK_STREAM, 0)) < 0) return -1;
        if(bind(s, (struct sockaddr *)&sa, sizeof(sa)) < 0){
                close(s);
                return -1;
                }

/*      Start listening on the TCP port    */

        listen(s, 3);
        return s;
        }
```

```
int get_connection(int s){
/*
        Get client connection
        Parameters: s: socket handler
        Return value: connection handler
*/

        struct sockaddr_in isa;
        unsigned long i;
        int t;

        i=sizeof(isa);
        getsockname(s, (struct sockaddr *)&isa, &i);

/*      Get and accept connection form clients    */

        if((t=accept(s, (struct sockaddr *)&isa, &i)) < 0){
                return -1;
                }
        return t;
        }


void waitchild(void){
/*      Wait for a child process to terminate */
        int wstatus;
        while(wait3(&wstatus,WNOHANG,NULL) > 0);
        }


void server(int t){
/*
        The server code. This function communicate with the clients.
        Parameters: t: client connection handler
*/
        char string[255];
        int c;
        FILE *f;

/*      Read a string from the client    */

        c=read(t, &string[0], 255);
        string[c]='\0';

/*      Open the data file     */

        f=fopen("/sharedfs/app/strings.data","a");
        if(f==NULL){

/*      In case of I/O error, write an error message to the STDERR    */

                fprintf(stderr,"Can not open strings.data file!\n");
                c=-1;

/*      send an error code to the client    */

                write(t,&c,sizeof(c));
                return;
                }

/*      Write the string into the data file    */

        fprintf(f,"%s\n",string);

/*      Send back to the client the number of bytes written out    */

        write(t,&c,sizeof(c));

/*      Close data file    */

        fclose(f);
        }


int main(int argc, const char * argv[])
{
        int s, t;

/*      Establish the socket connection and listen    */

        if((s=establish(PORT)) < 0) {
                perror("Can not establish socket connection");
                exit(1);
                }

/*      Set up SIGCHLD signal: in case of SIGCHLD run waitchild process  */

        signal(SIGCHLD, waitchild);

/*      Infinitive loop     */

        for (;;) {

/*      Check for client connection    */
```

```
                if((t=get_connection(s)) < 0) {
                        if(errno != EINTR) {

/*      TCP socket error    */

                                perror("Server connection error");
                                exit(1);
                                }
                        else continue;
                        }

/*      Now we have a client connected to the server, start server process
*/

                switch( fork() ) {
                        case -1 :

/*      Fork error    */

                                perror("Server error: fork");
                                close(s);
                                close(t);
                                exit(1);
                        case 0 :
/*      Start server process    */
                                server(t);
                                exit(0);
                        default :
                                close(t);
                                continue;
                        }
                }
        }
```

C.3 Cluster sense program

This sample cluster sense program is based on the clinfo C API. The program checks the status of the cluster with the given cluster ID. It also checks the status of the cluster nodes' network interfaces. You must install the cluster.es.client filesets to compile and run this program.

Program usage: `cl clusterID`

```
/*
cl.c
```

```
Sample Clinfo client program. It checks the status of the cluster. You must
specify the cluster ID as a parameter.
Usage: cl clusterID
Example: cl 77
Compile: cc -lcl -lclstr -o cl cl.c
*/

#include <stdio.h>
#include <stdlib.h>

/* Clinfo C API header files */
#include <sys/types.h>
#include <netinet/in.h>
#include <cluster/clinfo.h>


char get_state(int cl_state){
/*
        This function convert the enum cls_state variables to meaningful
        messages. We will use this function to print out the status of the
        cluster, cluster nodes and interfaces
*/
        switch(cl_state){
                case CLS_INVALID: printf("Invalid");
                        break;
                case CLS_VALID: printf("Valid");
                        break;
                case CLS_UP: printf("UP");
                        break;
                case CLS_DOWN: printf("DOWN");
                        break;
                case CLS_UNKNOWN: printf("Unknown");
                        break;
                case CLS_GRACE: printf("Grace");
                        break;
                case CLS_JOINING: printf("Joining");
                        break;
                case CLS_LEAVING: printf("Leaving");
                        break;
                case CLS_IN_USE: printf("In use");
                        break;
                case CLS_PRIMARY: printf("Primary");
                        break;
                default:
                        ;
                }
        }
```

```
char get_substate(int cl_substate){
/*
        This function convert the enum cls_substate variables to meaningful
        messages. We will use this function to print out the status of the
        cluster, cluster nodes and interfaces.
*/
        switch(cl_substate){
                case CLSS_UNKNOWN: printf("unknown substate");
                        break;
                case CLSS_STABLE: printf("stable");
                        break;
                case CLSS_UNSTABLE: printf("unstable");
                        break;
                case CLSS_ERROR: printf("error");
                        break;
                case CLSS_RECONFIG: printf("in reconfiguration");
                        break;
                default: ;
                }
        }


int main(char argc, char *argv[]){
        int i,j;
        int clusterid, status, nodes;
        struct cl_node *nodemap;
        struct cl_cluster clusterbuf;

/*      Parameter checking        */

        if(argc!=2){
        printf("Bad parameters! Usage: cl clusterid\n");
                exit(2);
                }
/*      Get the cluster ID form the command line parameter    */
        clusterid=atoi(argv[1]);
        if(clusterid==0){
                printf("Invalid cluster id! Usage: cl clusterid\n");
                exit(2);
                }

/*      Get the cluster status    */

        status=cl_getcluster(clusterid,&clusterbuf);
        if(status!=CLE_OK){
                cl_perror(status, "Can not get cluster information:");
                exit(2);
                }

/*      Print the cluster status    */

        printf("Cluster name: %s\n",clusterbuf.clc_name);
        printf("Cluster status: ");
        get_state(clusterbuf.clc_state);
        printf(" and ");
        get_substate(clusterbuf.clc_substate);
        printf("\n");

/*      Allocate memory for nodemap structure    */

        cl_alloc_nodemap(&nodemap);
        if(nodemap==0){
                printf("Memory allocation error!\n");
                exit(2);
                }

/*      Get the node and interface status information    */

        nodes=cl_getnodemap(clusterid, nodemap);
        if(nodes<0){
                printf("Can not get node information!\n");
                exit(2);
                }

/*      Print node information for each node    */

        for(i=0;i!=nodes;i++){
                printf("Node %s is ",nodemap[i].cln_nodename);
                get_state(nodemap[i].cln_state);
                if(nodemap[i].cln_nif!=0){

/*      Print interface status information for each interface on each node
*/

                        printf(" and has the following interfaces:\n");
                        for(j=0;j!=nodemap[i].cln_nif;j++){
                             printf("%s is ",nodemap[i].cln_if->cli_name);
                             get_state(nodemap[i].cln_if->cli_state);
                             printf("\n");
                             nodemap[i].cln_if++;
                            }
                        }
                }

/*      Free nodemap's memory    */

        cl_free_nodemap(nodemap);
        exit(0);
        }
```

C.4 TX client application shell script

```
#!/bin/ksh

APPDIR=/demodir# Location of client application
CLPRG=$APPDIR/c4# Client program
SVPRG=$APPDIR/s4# Server program
CLMTR=$APPDIR/cl# Cluster sensing program
APPSVR=ford# production server
BAKSVR=dodge# backup server
SVRCL=77# cluster ID
TMPFILE=/tmp/.cltmp
FLD="\$3"

pheader()
{
echo
"-------------------------------------------------------------------------
-----
                          TRANSACTION SERVICES CLIENT
-------------------------------------------------------------------------
----"
}


pause()
{
echo "Press Enter to Continue"
read pause
}


# MAIN PROGRAM


while true
do
pheader
```

```
# Checking that clinfo is active on the client node

if [ `lssrc -s clinfoES | cut -c44- | tail +2` != "active" ]
then
echo "Cluster information daemon is not active on this system"
echo "Please start the clinfo before you run this application"
exit
fi

# Checking cluster status

echo "Checking cluster status"
$CLMTR $SVRCL> $TMPFILE 2>&1
if [ $? -ne 0 ]
then
echo "The cluster is not active"
exit
else
TRANS=`date`
if [ `expr "$TRANS" : ".*"` -ne 0 ]
then
if [ "$TRANS" = "exit" ]
then
exit 0
fi
echo "Checking cluster status"
$CLMTR $SVRCL > $TMPFILE
APPSTAT=`awk "(/$APPSVR is/ || /${APPSVR}_boot is/) && !/and/ { print $FLD
}" $TMPFILE`
BAKSTAT=`awk "(/$BAKSVR is/ || /${BAKSVR}_boot is/) && !/and/ { print $FLD
}" $TMPFILE`
if [ $APPSTAT = "UP" ]
then

# submit transaction to the primary server

$CLPRG $APPSVR "$TRANS"
if [ $? -ne 0 ]
then
echo "Please verify that the application server is running on the node"
fi
else
if [ $BAKSTAT = "UP" ]
then

# submit transaction to the backup server
```

```
$CLPRG $BAKSVR "$TRANS"
if [ $? -ne 0 ]
then
echo "Please verify that the application server is running on the node"
fi
fi
fi
fi
fi
sleep 10
done
```

Appendix D. Using the additional material

This redbook also contains additional material. The data is available on the Internet as Web material. See the appropriate section below for instructions on using or downloading each type of material.

D.1 Using the downloadable Web material

The downloadable data that accompanies this redbook is packed into a zip file. The zip file contains all scripts and programs listed in Appendix A to C. The file size is 50 KB. To use the material, download the zip file. When it has downloaded, unzip the file and edit or copy the files you need. For ease of use, the files unpack into a directory structure that reflects the source appendix.

File name	*Description*
SG24-4498.ZIP	Code examples from appendices

D.1.1 System requirements for downloading and using Web material

The following system configuration is recommended for optimal use of the CD-ROM or diskette.

Hard disk space:	1 MB minimum
Operating System:	AIX 4.3.3 or higher
Processor:	Any RISC/6000
Memory:	No requirements
Other:	HACMP or HACMP/ES 4.3.1 or higher

D.2 Locating the additional material on the Internet

The Web material associated with this redbook is available in soft-copy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

`ftp://www.redbooks.ibm.com/redbooks/SG24-4498/SG24-4498.ZIP`

Alternatively, you can go to the IBM Redbooks Web site at:

`http://www.redbooks.ibm.com/`

Select the **Additional materials** and open the directory that corresponds with the redbook form number.

D.2.1 How to use the Web material

Create a subdirectory (folder) on your workstation and copy the contents of the Web material into this folder.

Appendix E. Special notices

This publication is intended to help systems engineers, service providers, and customers to successfully implement and customize HACMP installations. The information in this publication is not intended as the specification of any programming interfaces that are provided by HACMP Version 4.3.1. See the PUBLICATIONS section of the IBM Programming Announcement for IBM High Availability Cluster Multiprocessing (HACMP) for AIX for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM ("vendor") products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate

Appendix F. Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

F.1 IBM Redbooks

For information on ordering these publications see “How to get IBM Redbooks” on page 309.

- *Bullet-Proofing Your Oracle Database with HACMP: A Guide to Implementing AIX Databases with HACMP*, SG24-4788
- *HACMP Enhanced Scalability*, SG24-2081 (Available in softcopy format only)
- *HACMP Enhanced Scalability Handbook*, SG24-5328
- *HACMP Enhanced Scalability: User-Defined Events*, SG24-5327
- *High Availability and Scalability with Domino Clustering and Partitioning on AIX*, SG24-5163
- *High Availability Scenarios for Tivoli Software*, SG24-2032
- *IBM Certification Study Guide AIX HACMP*, SG24-5131
- *Informix Cluster POWERsolution Guide*, SG24-2020 (Available in softcopy format only)
- *Lotus Domino R5 for IBM RS/6000*, SG24-5138
- *Migrating to HACMP/ES*, SG24-5526
- *Oracle Cluster POWERsolution Guide*, SG24-2019 (Available in softcopy format only)
- *Problem Solving and Troubleshooting in AIX Version 4.3*, SG24-5496
- *PSSP 3.1 Announcement*, SG24-5332

F.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at http://www.redbooks.ibm.com/ for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	**Collection Kit Number**
System/390 Redbooks Collection	SK2T-2177

CD-ROM Title	**Collection Kit Number**
Networking and Systems Management Redbooks Collection	SK2T-6022
Transaction Processing and Data Management Redbooks Collection	SK2T-8038
Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
AS/400 Redbooks Collection	SK2T-2849
Netfinity Hardware and Software Redbooks Collection	SK2T-8046
RS/6000 Redbooks Collection (BkMgr)	SK2T-8040
RS/6000 Redbooks Collection (PDF Format)	SK2T-8043
Application Development Redbooks Collection	SK2T-8037
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

F.3 Other resources

These publications are also relevant as further information sources:

- *AIX V4.3 Installation Guide*, SC23-4112
- *AIX Versions 3.2 and 4 Performance Monitoring and Tuning Guide*, SC23-2365
- *AIX Version 4.3 Problem Solving Guide and Reference*, SC23-4123
- *HACMP V4.3 Concepts and Facilities*, SC23-4276
- *HACMP V4.3 Enhanced Scalability and Administration Guide Vol. 1 and Vol. 2*, SC23-4284
- *HACMP V4.3 AIX Planning Guide*, SC23-4277
- *HACMP V4.3 AIX: Programming Client Applications*, SC23-4282
- *HACMP V4.3 AIX: Troubleshooting Guide*, SC23-4280
- *IBM Enterprise Storage Server Introduction and Planning Guide*, GC26-7294
- *Lotus C API Toolkit for Domino and Notes Release 5.0.2* (Available from `http://www.lotus.com/home.nsf/welcome/developernetwork`)
- *Tivoli Storage Manager for AIX Administrator's Guide,* GC35-0368
- *Tivoli Storage Manager for AIX Quick Start,* GC35-0367
- *Tivoli Storage Manager for UNIX, Using the Backup-Archive*, SH26-4105

F.4 Referenced Web sites

These Web sites are also relevant as further information sources:

- `http://www.hursley.ibm.com/~ssa/rs6k/` Link to the SSA hot spare disk tool.
- `http://www.lotus.com/home.nsf/welcome/developernetwork` Link to Lotus developer page where the Domino C-API can be downloaded.
- `http://www.redbooks.obm.com`
- `http://www.tivoli.com/tsm app` Link to a list of devices supported by Tivoli TSM.

How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** http://www.redbooks.ibm.com/

  Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

  Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders by e-mail including information from the IBM Redbooks fax order form to:

	e-mail address
In United States or Canada	pubscan@us.ibm.com
Outside North America	Contact information is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM Intranet Web site at http://w3.itso.ibm.com/ and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at http://w3.ibm.com/ for redbook, residency, and workshop announcements.

IBM Redbooks fax order form

Please send me the following:

Title	Order Number	Quantity

First name	Last name	
Company		
Address		
City	Postal code	Country
Telephone number	Telefax number	VAT number

☐ Invoice to customer number

☐ Credit card number

Credit card expiration date	Card issued to	Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Glossary

AIO.	Asynchronous I/O.
AIX.	Advanced Interactive Executive.
API.	Application Programming Interface.
ARP.	Address Resolution Protocol.
ASCII.	American Standard Code for Information Interchange.
AS/400.	Application System/400.
CD-ROM.	Compact Disk - Read Only Memory.
CLVM.	Concurrent Logical Volume Manager.
C-SPOC.	Cluster-Single Point of Control.
DARE.	The Dynamic Reconfiguration (DARE) Resource Migration utility.
DE.	Differential Ended. An SCSI standard.
DLC.	Data Link Control.
DMS.	HACMP Dead Man Switch. This program is always running in an HACMP system and has to be reset at intervals; otherwise, the DMS will stop the system.
ESS.	Enterprise Storage Server.
GB.	Gigabyte.
GODM.	Global Object Data Manager.
GUI.	Graphical User Interface.
HACMP/6000.	High Availability Cluster Multi-Processing/6000.
HACMP/ES.	High Availability Cluster Multi-Processing/Enhanced Scalability.
HANFS.	High Availability Network File System.
HCON.	Host Connection Program.
IBM.	International Business Machines Corporation.
I/O.	Input/Output.
IP.	Interface Protocol.
IPAT.	IP Address Takeover.
ITSO.	International Technical Support Organization.
KB.	Kilobyte.
kb.	kilobit.
LAN.	Local Area Network.
LP.	Logical Partions.
LVM.	Logical Volume Manager.
MAC.	Medium Access Control.
MB.	Megabyte.
NETBIOS.	Network Basic Input/Output System.
NFS.	Network File System.
NIM.	Network Interface Module.
NIS.	Network Information Service.
ODM.	Object Data Manager.
PAD.	Packet Assembler/Disassembler.
PPRC.	Peer-to-Peer Remote Copy.
PTF.	Program Temporary Fix.
RAID.	Redundant Array of Independent Disks.
RAN.	Remote Asynchronous Node.
RISC.	Reduced Instruction Set Computer.
SCSI.	Small Computer Systems Interface.
SCSI-2.	An enhanced version of SCSI that increases performance by adding features and commands to the SCSI standard.
SDR.	System Data Repository.
SID.	System ID.
SMIT.	System Management Interface Tool.
SP.	IBM Scalable Parallel systems family of AIX systems.
SPX/IPX.	Sequenced Package Exchange/Internetwork Packet Exchange.
SRC.	System Resource Controller.
SSA.	Serial Storage Architecture.
SVC.	Switched Virtual Circuit.

syncd. AIX syncd is a program that always runs on AIX systems and handles the synchronization of data to disk.

TCP/IP. Transmission Control Protocol/Interface Protocol.

VGDA. Volume Group Descriptor Area.

WAN. Wide Area Network.

Index

IBM Redbooks review

Your feedback is valued by the Redbook authors. In particular we are interested in situations where a Redbook "made the difference" in a task or problem you encountered. Using one of the following methods, **please review the Redbook, addressing value, subject matter, structure, depth and quality as appropriate.**

- Use the online **Contact us** review redbook form found at **ibm.com**/redbooks
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Document Number **Redbook Title**	SG24-4498-01 HACMP/ES Customization Examples
Review	
What other subjects would you like to see IBM Redbooks address?	
Please rate your overall satisfaction:	O Very Good O Good O Average O Poor
Please identify yourself as belonging to one of the following groups:	O Customer O Business Partner O Solution Developer O IBM, Lotus or Tivoli Employee O None of the above
Your email address: The data you provide here may be used to provide you with information from IBM or our business partners about our products, services or activities.	O Please do not use the information collected here for future marketing or promotional contacts or other communications beyond the scope of this transaction.
Questions about IBM's privacy policy?	The following link explains how we protect your personal information. **ibm.com**/privacy/yourprivacy/

IBM

Redbooks

HACMP/ES Customization Examples

HACMP/ES Customization Examples

Configure Lotus Domino servers for high availability

Program cluster-aware applications

Detailed, step-by-step examples

HACMP and HACMP/ES are the IBM tools for building AIX-based, mission-critical computing platforms. HACMP Enhanced Scalability or HACMP/ES is a more scalable and strategic version of the standard HACMP version. IBM HACMP and HACMP Enhanced Scalability provide high availability for RS/6000 systems through adding software and redundant hardware components.

This IBM Redbook is an essential guide to configuring RS/6000 and HACMP in various customer situations. This book delivers practical, tested information on how to customize HACMP and tailor the configuration for the specific customer needs. Detailed sample application programs and in-depth, step-by-step examples help you learn how to take advantage of the full functionality of HACMP/ES. Included are examples that describe how to configure Lotus Domino for high availability and how to program a cluster-aware application where a client application checks the cluster status before requesting a transaction.

This comprehensive guide to customizing HACMP/ES will provide system engineers and customers the technical insight necessary to design and successfully implement customizations in an HACMP cluster.

SG24-4498-01 ISBN 0738417130