High availability SAP Adaptive Server Enterprise (ASE) on IBM AIX using IBM PowerHA

An IBM and SAP white paper on SAP ASE 15.7 with PowerHA for AIX 7.1 on IBM POWER7 processor-based servers

Deepak Narayana (IBM), Jian Yang (SAP), Peter H Barnett (IBM), Shawn Bodily (IBM)

IBM Systems and Technology Group ISV Enablement

December 2013

@IBMSystemsISVs

Table of contents

Abstract

This paper is jointly created by IBM and SAP to assist database administrators (DBAs) and UNIX system administrators on setting up a high availability (HA) solution for SAP Adaptive Server Enterprise (ASE) 15.7 on IBM AIX 7.1 on IBM POWER7 processor-based hardware using the IBM PowerHA solution.

Introduction

This paper's primary focus is on businesses that are implementing the high availability solution for SAP ASE instances on IBM® AIX® in one of the following environments:

- Implementing new SAP ASE instances on IBM Power Systems™ with IBM AIX 7.1
- Upgrading from earlier IBM Power Systems and AIX installations
- Migrating to an IBM Power Systems server running AIX 7.1 from another version of the UNIX® operating system

Note: The operating system is referred to as AIX. The intermediate release levels, called technology levels, are abbreviated as **TL** where referenced. SAP ASE 15.7 is referred to as ASE unless a feature or an issue is specific to a certain level.

The scope of this white paper is to demonstrate the interfacing technique between SAP ASE and IBM PowerHA® on AIX.

SAP ASE manuals address general ASE installations and configuration.
http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc35892.1570/doc/pdf/aseigibm.pdf

PowerHA manuals address the system requirements and physical hardware setups.
ibm.com/redbooks/redpieces/abstracts/sg248106.html
PowerHA website - **ibm.com**/systems/power/software/availability/

AIX and IBM Power Systems manuals and white papers address details about the server, the operating system, storage, and virtualization that are generic to database servers and are beyond the scope of this paper. Refer to the following links for more details.

- **ibm.com**/systems/power/software/aix/resources.html
- **ibm.com**/systems/power/

The focus of this white paper is also on the aspect of database administration in which the DBA and UNIX system administrators interact closely. Ideally, DBAs who read this paper might be competent in SAP ASE; UNIX system administrators might be competent in IBM Power Systems, AIX, and storage. Both should be reasonably acquainted with the other’s specialty. There are some commands that require UNIX **root** privileges. Other commands require the SAP ASE **sa_role** authority.

Logical partitions (LPARs) and dynamic logical partitions (DLPARs)

All IBM POWER5™, IBM POWER6®, and POWER7 processor-based UNIX servers are LPAR-capable. That is, they support LPARs, which are multiple OS images of varying sizes and capacities that exist on a single physical server frame. Even if there is only one image on an applicable hardware platform, it is still an LPAR.

The fore mentioned IBM POWER® processor-based servers support the dynamic reallocation of memory and processors, referred to as DLPARs. The system administrator can add or remove processors and memory in an LPAR without restarting. This assumes that the chosen values stay within the minimum and maximum memory and processor values specified in the LPAR profile. The same applies to virtual resources, which are discussed in the following sections.

Environment

This section describes the hardware and software environment used for this exercise. The configuration might vary depending on each end-user's requirement.

Test setup

PowerHA can be configured using various configurations of LPAR and storage. Some examples (but not limited to) are provided in the following list

- LPARs on same physical systems sharing local disks [direct attach, virtual Small Computer System Interface (VSCSI), or N-Port ID Virtualization (NPIV)]. This is excellent for a test environment, but not recommended for production.
- LPARs on multiple physical system sharing external disks [direct attach, vSCSI, or Fibre Channel (FC) based NPIV]
- LPARs at remote location using IBM General Parallel File System (IBM GPFS™) or shared external disks through storage area network (SAN).

The test environment consisted of the following hardware and software.

Hardware:

- Two IBM Power® 770 servers
  - One LPAR on each Power 770 server with:
    - Four processors
    - 16 GB memory
    - Local disk for OS (AIX) and SAP ASE binary installations
- Three shared IBM Storwize V7000 logical unit numbers (LUNs), accessed through NPIV
  - Two are used for each of two SAP instances

    - One to be used for PowerHA cluster repository disk

Software:

- AIX 7.1 TL2 SP2 (64-bit w/SMT4 enabled)
- IBM PowerHA SystemMirror for AIX v7.1.2
- SAP ASE 15.7

Each of the two LPARs will host a separate database instance and also participate in a PowerHA cluster.

General recommendation

In an HA environment, its generally recommended to have a root volume group (rootvg) on either a mirrored disk or Redundant Array of Independent Disks (RAID) that is, 1,5,6,10 disks. This can be attained by using Logical Volume Manager (LVM) mirroring using the RAID features in storage subsystems or through Virtual I/O Server (VIOS). Also, attention must be paid to I/O adapters (both virtual and physical, if any) on the LPAR to ensure redundant path and multipath I/O (MPIO) options. Although PowerHA can handle OS / hardware failure, it is best to eliminate single points of failure (SPOFs) to prevent unnecessary failovers from occurring.

HA setup is best if set up on two separate physical hardware and a dual-VIOS environment within each server is strongly recommended. Refer to the PowerVM best practice guide at:
ibm.com/redbooks/redpapers/pdfs/redp4194.pdf

PowerHA SystemMirror on AIX

PowerHA is an IBM product that provides high availability, business continuity, disaster recovery solutions on AIX. This cluster product helps to provide near continuous businesses critical application availability, where there is an increasing demand for 24x7 availability. It can monitor hardware, OS, and applications and provide automated failover options when failures are detected.

Always ensure that PowerHA is installed with latest Service Packs available from IBM Fix Central.

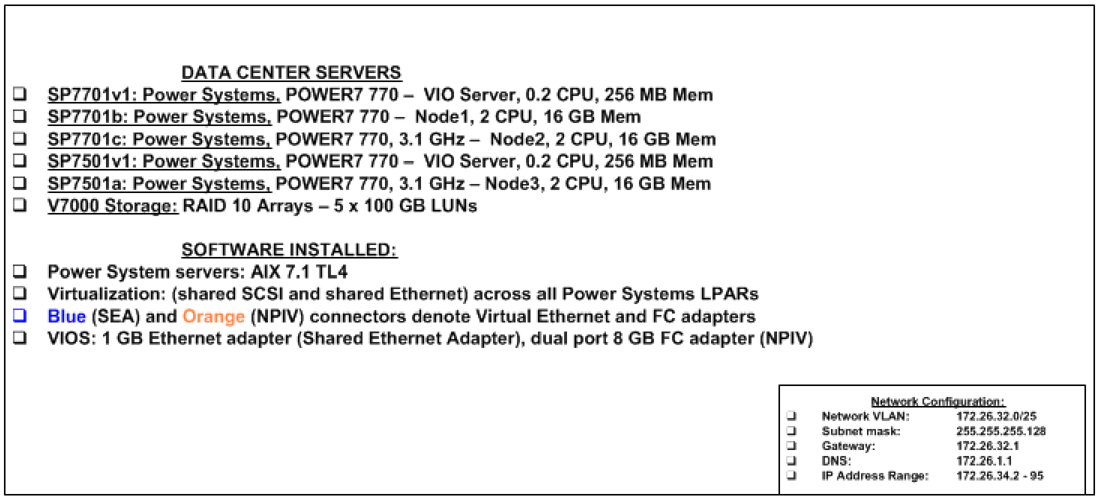
Prerequisites

Refer to SAP ASE and PowerHA manuals for software prerequsisites. It is recommended to use IBM AIX Standard or Enterprise editions.

- Make sure that there is adequate paging space and free space on the /tmp and / (root) file systems.
- Verify the ulimits setting for each of the products.
- Make a note of the Service IP and System IP for the two hosts. Verify whether the Domain Name System (DNS) entries are valid and host name resolves properly. Keep the /etc/hosts files up-to-date on both systems.
- Refer to the tuning guide and best practices guide.
- Verify whether the available free shared disks are enabled for concurrent access on both nodes.
- Create the /etc/cluster/rhosts file:
  1. Enter only the IP address that resolve to the host name of each system/cluster node.
  2. Copy the file to all nodes.
  3. Restart the clcomd daemon on all nodes using *refresh -s clcomd*.

Test topology

The following figure shows the setup used for this exercise.



DATA CENTER SERVERS

- ❑ SP7701v1: Power Systems, POWER7 770 – VIO Server, 0.2 CPU, 256 MB Mem
- ❑ SP7701b: Power Systems, POWER7 770 – Node1, 2 CPU, 16 GB Mem
- ❑ SP7701c: Power Systems, POWER7 770, 3.1 GHz – Node2, 2 CPU, 16 GB Mem
- ❑ SP7501v1: Power Systems, POWER7 770 – VIO Server, 0.2 CPU, 256 MB Mem
- ❑ SP7501a: Power Systems, POWER7 770, 3.1 GHz – Node3, 2 CPU, 16 GB Mem
- ❑ V7000 Storage: RAID 10 Arrays – 5 x 100 GB LUNs

SOFTWARE INSTALLED:

- ❑ Power System servers: AIX 7.1 TL4
- ❑ Virtualization: (shared SCSI and shared Ethernet) across all Power Systems LPARs
- ❑ Blue (SEA) and Orange (NPIV) connectors denote Virtual Ethernet and FC adapters
- ❑ VIOS: 1 GB Ethernet adapter (Shared Ethernet Adapter), dual port 8 GB FC adapter (NPIV)

Network Configuration:

❑	Network VLAN:	172.26.32.0/25
❑	Subnet mask:	255.255.255.128
❑	Gateway:	172.26.32.1
❑	DNS:	172.26.1.1
❑	IP Address Range:	172.26.34.2 - 95

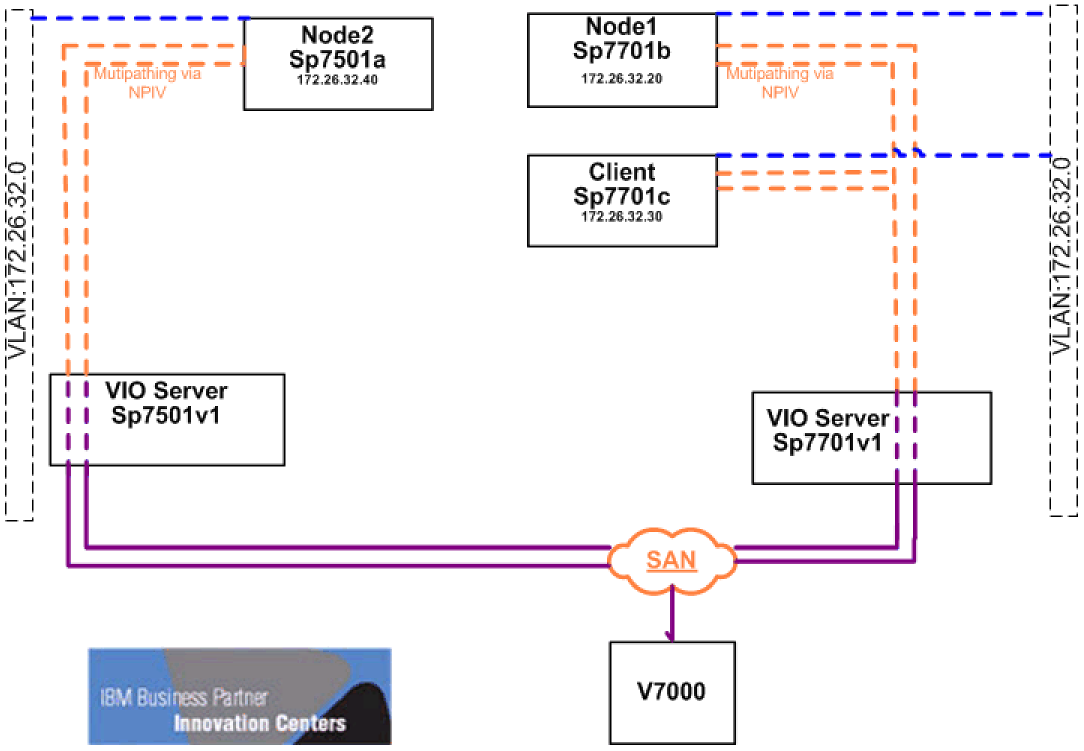


Figure 1: Test topology

Configuring PowerHA

PowerHA can be configured using command line, System Management Interface Tool (SMIT), or IBM Systems Director. In this exercise, SMIT panels are used to configure.

SMIT panels for PowerHA can be accessed using the `smitty sysmirror` command.

First, you need to define the shared disks for the two ASE instances. Then create two volume groups, sybasevg1 and sybasevg2, on each LPAR.

The LPAR host names that have been defined are:

- sp7501a (hereafter referred to as Node A)
- sp7701b (hereafter referred to as Node B)

Shared storages for ASE

You need to identify two shared disks for ASE from the available disks on the system.

```
hdisk0          00f6081590bb198e                    rootvg          active
hdisk1          00f60815fb20f5c0                    None
hdisk2          00f60815fd2b8ee7                    None
hdisk3          00f60815fd2b8d75                    None
```

Make sure they display same Port VLAN ID (PVID) on both LPARs. The hdisk numbers do not have to match, but it is common that they usually do.

Log in to Node A and define a shared disk for ASE using the following command:

```
# mkvg -y sybasevg1 -f -n -C hdisk1
```

(Note: The `-C` option is required to create an enhanced, concurrent, and capable volume group.)

Create a file system on the volume group using the following command

```
# crfs -v jfs2 -g sybasevg1 -a size=20G -m /sybase1 -a logname=INLINE
```

It is recommended to create the logical volumes (LVs) before creating a file system to have better control on naming in HA configuration.

Now, you need to import this definition on Node B to make it aware of the newly created LVM settings.

First, vary off the volume group using the following command.

```
# varyoffvg sybasevg1
```

Next, log in to Node B and import the volume group using the following command.

```
# importvg -n -y sybasevg1 hdisk1
```

Repeat these steps for a second shared ASE disk naming the volume group *sybasevg2* on hdisk2 and create a file system, named */sybase2*, on Node B. Then import it to Node A.

Configuring clusters

We define clusters where the nodes and resources would be specified.

a) Define a cluster

Now, define the main cluster and add the nodes to the cluster. We use smitty panels and start with Node A.

```
# smitty sysmirror
```

Click **Custom Cluster Configuration → Cluster Nodes and Networks → Initial Cluster Setup (Custom) → Cluster → Add/Change/Show Cluster**.

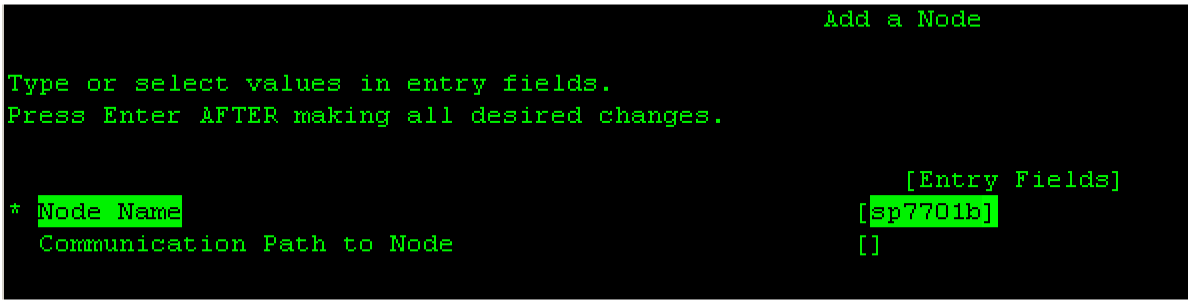
Then type the cluster name and press **Enter**.

b) Define nodes

Now, add the nodes to this cluster that you defined.

```
# smitty sysmirror
```

Click **Custom Cluster Configuration → Cluster Nodes and Networks → Initial Cluster Setup (Custom) → Nodes → Add a Node**.



```
                                                                      Add a Node

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

                                                                            [Entry Fields]
* Node Name                                                              [sp7701b]
  Communication Path to Node                                             []
```

Figure 2: Add a node

Similarly, add the other host/Node B as well to the cluster using the smitty panel on Node A.

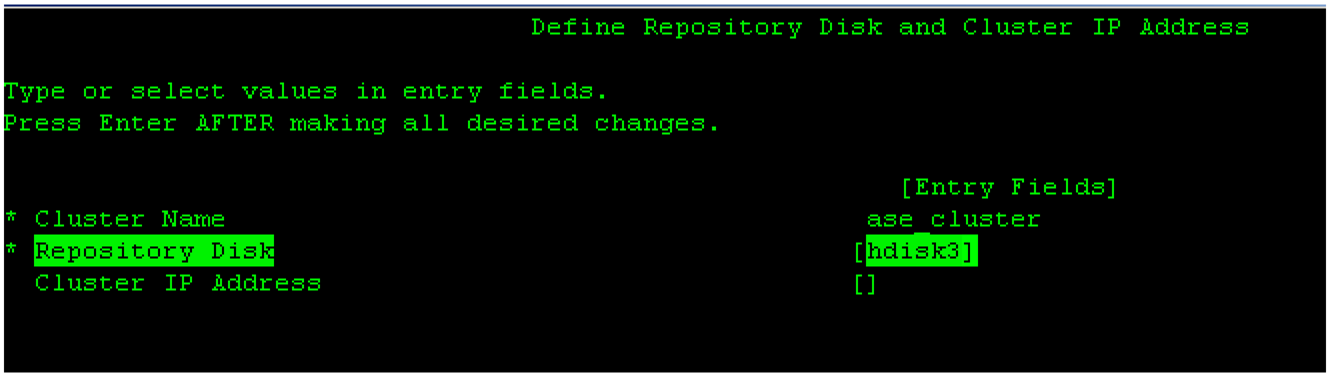
The above two steps (a and b) can also be done in one step using the **Cluster Nodes and Networks → Standard Cluster Deployment → Setup a Cluster, Nodes and Networks** option.

c) Define the cluster repository disk

Now, define the cluster repository disk.

```
# smitty sysmirror
```

Click **Custom Cluster Configuration → Cluster Nodes and Networks → Initial Cluster Setup (Custom) → Define Repository Disk and Cluster IP Address**.



```
                              Define Repository Disk and Cluster IP Address

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

                                                       [Entry Fields]
* Cluster Name                                        ase_cluster
* Repository Disk                                    [hdisk3]
  Cluster IP Address                                 []
```

Figure 3: Defining Repository Disk and Cluster IP Address

You might use the `cltopinfo` utility to check on the configuration for the cluster so far.

d) Syncronize the cluster

```
scripts # smitty sysmirror
```

Click **Cluster Applications and Resources → Verify and Synchronize Cluster Configuration**. This might take a few minutes.

e) Define application scripts

Next, you need to define the application controller scripts to start and stop ASE. Refer to ASE Start/Stop sample script provided, review, and make modifications accordingly. The ASE script used here is an original derivation from IBM HACMP™. There is one script which starts, stops, monitors and fails over. Parameters passed to it determine what it performs. The start script is passed with a monitor parameter. This script starts and also monitors the ASE instance, calling for a cluster failover if the DB fails. Hence, the PowerHA application monitoring feature is not used in this test. When a node_down is invoked, the failover of the DB occurs. Make sure that the scripts are available in the same location on both the nodes.

```
# smitty sysmirror
```

Click **Cluster Applications and Resources → Resources → Configure User Applications (Scripts and Monitors) → Application Controller Scripts → Add Application Controller Scripts**.

f) Define service IP

Ideally, this IP needs to be on its own dedicated network card.

```
# smitty sysmirror
```

Click **Cluster Applications and Resources → Resources → Configure Service IP Labels/Addresses → Add a Service IP Label/Address**. Then, select the network to use and proceed.

g) Define a resource group

```
# smitty sysmirror
```

Click **Cluster Applications and Resources → Resources → Add a Resource Group**.

Provide a name for the resource goup and select the nodes participating in this resource group in the order of priority. You can use the **F4** key to list all the nodes and select using the **F7** key. Choose the appropriate failover and fallback options. It is considered a best practice to use the fallback option of **Never Failback**.

Note: The following figure shows the default values.

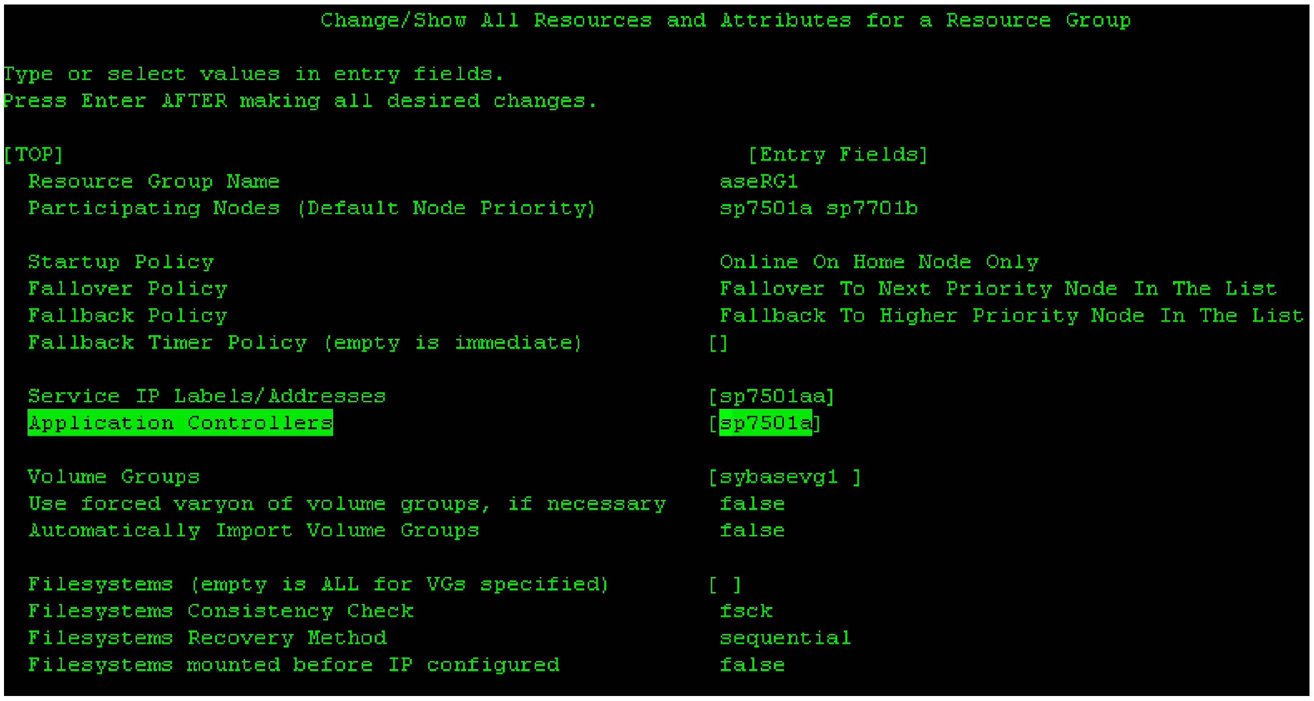
h) Updating the resource group

Finally we put all the resources previously defined into the resource group.

```
# smitty sysmirror
```

Click **Cluster Applications and Resources → Resources → Change/Show Resources and Attributes for a Resource Group**.

Select the service IP labels / addresses, application controllers, and volume groups in this panel and assign it to the resource group. When selecting each field, you can press **F4** to choose from a list of the created resources, making it unnecessary to manually type in the names.



```
                        Change/Show All Resources and Attributes for a Resource Group

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

[TOP]                                                              [Entry Fields]
  Resource Group Name                                             aseRG1
  Participating Nodes (Default Node Priority)                     sp7501a sp7701b

  Startup Policy                                                  Online On Home Node Only
  Fallover Policy                                                 Fallover To Next Priority Node In The List
  Fallback Policy                                                 Fallback To Higher Priority Node In The List
  Fallback Timer Policy (empty is immediate)                     []

  Service IP Labels/Addresses                                    [sp7501aa]
  Application Controllers                                        [sp7501a]

  Volume Groups                                                  [sybasevg1 ]
  Use forced varyon of volume groups, if necessary                false
  Automatically Import Volume Groups                              false

  Filesystems (empty is ALL for VGs specified)                   [ ]
  Filesystems Consistency Check                                   fsck
  Filesystems Recovery Method                                     sequential
  Filesystems mounted before IP configured                        false
```

Figure 4: Resource groups

i) Synchronizing the cluster

Before proceeding any further, you need to synchronize the setting across the nodes.

```
# smitty sysmirror
```

Click **Cluster Applications and Resources → Verify and Synchronize Cluster Configuration**. This might take a few minutes.

The Sync output might be displayed as follows:

```
Verification to be performed on the following:
        Cluster Topology
        Cluster Resources


Retrieving data from available cluster nodes.  This could take a few
minutes.


        Start data collection on node Sp7501a
        Start data collection on node Sp7701b
        Collector on node Sp7701b completed
        Collector on node Sp7501a completed
        Data collection complete


Verifying Cluster Topology...


        Completed 10 percent of the verification checks


Verifying Cluster Resources...


        Completed 40 percent of the verification checks


WARNING: Application monitors are required for detecting application
failures
in order for PowerHA SystemMirror to recover from them.  Application
monitors are started
by PowerHA SystemMirror when the resource group in which they
participate is activated.
The following application(s), shown with their associated resource
group,
do not have an application monitor configured:
Application Server                Resource Group
  -------------------------------  ---------------------------------
   aseApp                             aseRG
        Completed 50 percent of the verification checks
        Completed 60 percent of the verification checks
        Completed 70 percent of the verification checks
        Completed 80 percent of the verification checks

    .

    .

    .

Remember to redo automatic error notification if configuration has
changed.
Committing any changes, as required, to all available nodes...
Adding any necessary PowerHA SystemMirror entries to /etc/inittab and
/etc/rc.net for IPAT on node Sp7501a.
```

```
Adding any necessary PowerHA SystemMirror entries to /etc/inittab and
/etc/rc.net for IPAT on node Sp7701b.

Verification has completed normally.
```

Repeat steps d through i to create another resource group with Node B as the primary node.

SAP ASE setup and configurations

ASE can be configured mostly using the **sp** (stored procedure) commands. For others, such as interfaces file, needs to be modifie d manually. In this paper, `isql` commands are used to configure ASE.

Store procedure commands can be run in the `isql` client.

`“isql -Usa -P<sa password> -S<Server Name>”`

The test team installed an ASE server on both nodes. ASE install location was on a local disk. Master and system device files are installed on a shared disk.

Node A:

ASE installation: /sybase/ASE157

ASE devices: /sybase1/ASE157/master.dat & /sybase1/ASE157/sproc.dat

Node B:

ASE installation: /sybase/ASE157

ASE devices: /sybase2/ASE157/master.dat & /sybase2/ASE157/sproc.dat

Setting ASE scripts permission

Root permission is required for running $SYBASE/ASE-15_0/bin/sybha and $SYBASE/ASE-15_0/install/sybhauser scripts. If running as a non-root user, make sure that the user has root privileges.

For a non-root user:

```
# chown root sybha

# chgrp sybhagrp sybha

# chmod 4550 sybha

# chown root sybhauser

# chmod 600 sybhauser
```

Build ASE using resource files

To build a brand new ASE server for the PowerHA failover, you can use $SYBASE/ASE-15_0/bin/srvbuildres. If you already have configured the ASE server, change the configurations accordingly.

Node A:

Run `srvbuildres -r./ha.rs1`

ha.rs1:

```
sybinit.release_directory: /sybase/ASE157

sybinit.product: sqlsrv

sqlsrv.server_name: sp7501a

sqlsrv.sa_password: sybr3iic

sqlsrv.new_config: yes

sqlsrv.do_add_server: yes

sqlsrv.network_protocol_list: tcp

sqlsrv.network_hostname_list: sp7501aa

sqlsrv.network_port_list: 5000

sqlsrv.server_page_size: USE_DEFAULT

sqlsrv.force_buildmaster: no

sqlsrv.master_device_physical_name: /sybase1/ASE157/master.dat

sqlsrv.master_device_size: 800

sqlsrv.master_database_size: 300

sqlsrv.errorlog: USE_DEFAULT

sqlsrv.do_upgrade: no

sqlsrv.sybsystemprocs_device_physical_name: /sybase1/ASE157/sproc.dat

sqlsrv.sybsystemprocs_device_size: 800

sqlsrv.sybsystemprocs_database_size: 300
```

Node B:

Run `srvbuildres -r./ha.rs2`

ha.rs2:

```
sybinit.release_directory: /sybase/ASE157

sybinit.product: sqlsrv

sqlsrv.server_name: sp7701b

sqlsrv.sa_password: sybr3iic

sqlsrv.new_config: yes

sqlsrv.do_add_server: yes

sqlsrv.network_protocol_list: tcp

sqlsrv.network_hostname_list: sp7701bb

sqlsrv.network_port_list: 6000

sqlsrv.server_page_size: USE_DEFAULT

sqlsrv.force_buildmaster: no

sqlsrv.master_device_physical_name: /sybase2/ASE157/master.dat

sqlsrv.master_device_size: 800

sqlsrv.master_database_size: 300

sqlsrv.errorlog: USE_DEFAULT

sqlsrv.do_upgrade: no

sqlsrv.sybsystemprocs_device_physical_name: /sybase2/ASE157/sproc.dat

sqlsrv.sybsystemprocs_device_size: 800

sqlsrv.sybsystemprocs_database_size: 300
```

Edit ASE interfaces files

Modification of interfaces files for this exercise reflecting primary and failover nodes are as shown in this section.

Node A:

```
sp7501a
        master tcp ether sp7501aa 5000

        query tcp ether sp7501aa 5000

        hafailover sp7701b
```

```
Notes: sp7501aa is the cluster service IP on Node A.
```

Node B:

```
   sp7701b

           master tcp ether sp7701bb 6000

           query tcp ether sp7701bb 6000

           hafailover sp7501a

   Notes:  sp7701bb is the cluster service IP on Node B.
```

sp commands to set up ASE HA

ISQL interface can be used to configure HA by using the sp commands. This section shows the commands used in this exercise

1) Enable ASE HA on both nodes

```
1> sp_configure 'enable HA',1

2> go
```

Configuration option changed. Because the option is static, Adaptive Server must be restarted in order for the change to take effect. Changing the value of 'enable HA' to '1' increases the amount of memory that ASE uses by 264 KB.

Restart ASE.

2) Add primary and secondary servers

Node A:

```
1> sp_addserver sp7501a,local,sp7501a

2> go


1> sp_addserver sp7701b

2> go
```

Node B:

```
1> sp_addserver sp7701b,local,sp7701b

2> go


1> sp_addserver sp7501a

2> go
```

3) Perform other configurations on both nodes

```
1> sp_role "grant", ha_role, sa
2> go

1> sp_configure "enable xact coordination"
2> go

1> sp_addthreshold "master", "logsegment", 250, sp_thresholdaction
2> go
```

4) Verify ASE HA is enabled

```
1> sp_configure 'enable HA'
2> go
```

HA run value should be 1.

Run the ASE installhasvss script on both nodes

Running the HA script updates the ASE system databases and installs the ASE store procedures to be used by ASE HA failover.

```
# isql -Usa -Psybr3iic -Ssp7501a < $SYBASE/ASE-15_0/scripts/installhasvss
Configuration option changed. ASE need not be rebooted since the option is dynamic.
Changing the value of 'allow updates to system tables' does not increase the amount of memory Adaptive Server uses.
(return status = 0)
Installhasvss installation is complete.
(return status = 0)
```

Modify ASE HA scripts

The ASE HA script is located at $SYBASE/ASE-15_0/install/ASE_HA.sh. The script needs to be modified for the PowerHA failover environment.

Node A:

Copy $SYBASE/ASE-15_0/install/ASE_HA.sh to /usr/sbin/cluster/events/RUNHA_sp7501a.sh.

Modify RUNHA_sp7501a.sh to the following:

```
#
# Monitoring variables
#
MONITOR_INTERVAL=30
RECOVERY_TIMEOUT=300
SHUTDOWN_TIMEOUT=30
CONNECTION_TIMEOUT=120
RESPONSE_TIMEOUT=60
RETRY=0
ASE_FAILOVER=yes
BASIC_FAILOVER=yes



#
# General environment settings
#
export SYBASE_ASE=ASE-15_0
export SYBASE_OCS=OCS-15_0
export PATH=/sbin:/usr/sbin:/usr/bin
HA_LOGIN=
HA_PWD=
CIPHER=UXk2qEA-9HpgtXftUvyr9vDFz6qoFa-
gla8=bDYqr0nKUZWC4ux9..i61Ss75tsZgZT+8fK2HTR6Z6pHny@C3W@wsJi6GzPezrBWY35Wn@
#Notes: CIPHER option is to hide user password for best practise. Password
encryption can be done using ASE tool: $SYBASE/ASE-15_0/haisql
#If we don't use password encryption, then set  HA_LOGIN=sa and
HA_PWD=sybr3iic, leave the CIPHER field empty.

# Variables for Primary Companion
#
PRIM_SERVER=sp7501a
PRIM_HOST=sp7501a
PRIM_SYBASE=/sybase/ASE157
PRIM_SYBASE_HOME=${PRIM_SYBASE}/${SYBASE_ASE}
PRIM_ISQL=${PRIM_SYBASE}/${SYBASE_ASE}/bin/haisql
PRIM_CONSOLE_LOG=${PRIM_SYBASE_HOME}/install/${PRIM_SERVER}.cs_log
PRIM_RUNSCRIPT=${PRIM_SYBASE_HOME}/install/RUN_${PRIM_SERVER}

#
# Variables for Secondary Companion
#
SEC_SERVER=sp7701b
SEC_HOST=sp7701b
SEC_SYBASE=/sybase/ASE157
SEC_SYBASE_HOME=${SEC_SYBASE}/${SYBASE_ASE}
SEC_ISQL=${SEC_SYBASE}/${SYBASE_ASE}/bin/haisql
SEC_CONSOLE_LOG=${SEC_SYBASE_HOME}/install/${SEC_SERVER}.cs_log
```

Node B:

Copy $SYBASE/ASE-15_0/install/ASE_HA.sh to /usr/sbin/cluster/events/RUNHA_sp7701b.sh.

Modify RUNHA_sp7701b.sh to the following:

RUNHA_sp7701b.sh:

```
#
# Monitoring variables
#
MONITOR_INTERVAL=30
RECOVERY_TIMEOUT=300
SHUTDOWN_TIMEOUT=30
CONNECTION_TIMEOUT=120
RESPONSE_TIMEOUT=60
RETRY=0
ASE_FAILOVER=yes
BASIC_FAILOVER=yes

#
# General environment settings
#
export SYBASE_ASE=ASE-15_0
export SYBASE_OCS=OCS-15_0
export PATH=/sbin:/usr/sbin:/usr/bin
HA_LOGIN=
HA_PWD=
CIPHER=UXk2qEA-9HpgtXftUvyr9vDFz6qoFa-
gla8=bDYqr0nKUZWC4ux9..i61Ss75tsZgZT+8fK2HTR6Z6p
Hny@C3W@wsJi6GzPezrBWY35Wn@
#Notes: CIPHER option is to hide user password for best practise.
Password encryption can be done using ASE tool: $SYBASE/ASE-
15_0/haisql
If we don't use password encryption, then set  HA_LOGIN=sa and
HA_PWD=sybr3iic, leave the CIPHER field empty.


#
# Variables for Primary Companion
#
PRIM_SERVER=sp7701b
PRIM_HOST=sp7701b
PRIM_SYBASE=/sybase/ASE157
PRIM_SYBASE_HOME=${PRIM_SYBASE}/${SYBASE_ASE}
PRIM_ISQL=${PRIM_SYBASE}/${SYBASE_ASE}/bin/haisql
PRIM_CONSOLE_LOG=${PRIM_SYBASE_HOME}/install/${PRIM_SERVER}.cs_log
PRIM_RUNSCRIPT=${PRIM_SYBASE_HOME}/install/RUN_${PRIM_SERVER}
```

```
#
# Variables for Secondary Companion
#
SEC_SERVER=sp7501a
SEC_HOST=sp7501a
SEC_SYBASE=/sybase/ASE157
SEC_SYBASE_HOME=${SEC_SYBASE}/${SYBASE_ASE}
SEC_ISQL=${SEC_SYBASE}/${SYBASE_ASE}/bin/haisql
SEC_CONSOLE_LOG=${SEC_SYBASE_HOME}/install/${SEC_SERVER}.cs_log
```

Set up both ASE servers as companion servers

The **Symmetric normal** status of a companion server means that either node can fail over to the other node in a failover event.

Node A:

```
1> sp_companion "sp7701b", configure, null, sa, null
2> go

Server 'sp7501a' is alive and cluster configured.
Step: Access verified from Server:'sp7501a' to Server:'sp7701b'.
Server 'sp7701b' is alive and cluster configured.
Step: Access verified from Server:'sp7701b' to Server:'sp7501a'.
Step: Companion server's configuration check succeeded.
Step: Server handshake succeeded.
Step: Master device accessible from companion.
Step: Added the servers 'sp7501a' and 'sp7701b' for cluster configuration.
Step: Server configuration initialization succeeded.
Step: Synchronizing Application Specific information from companion server
Step: Synchronizing Roles from companion server
Step: Synchronizing Login Roles from companion server
Step: Synchronizing Remote Logins from companion server
Step: Synchronizing Groups in sysusers from companion server
Step: Synchronizing Sysattributes from companion server
Step: Synchronizing server logins from companion server
Step: Synchronizing server-wide privs from companion server
Step: User information synchronization succeeded.
Step: Server configured in normal companion mode.
(return status = 0)
```

```
1> sp_companion
2> go
Server 'sp7501a' is alive and cluster configured.
Server 'sp7501a' is configured for HA services.
Server 'sp7501a' is currently in 'Secondary normal' mode.
(return status = 0)
```

Node B:

```
1> sp_companion "sp7501a", configure, null, sa, null
2> go

Server 'sp7701b' is alive and cluster configured.
Step: Access verified from Server:'sp7701b' to Server:'sp7501a'.
Server 'sp7501a' is alive and cluster configured.
Step: Access verified from Server:'sp7501a' to Server:'sp7701b'.
Step: Companion server's configuration check succeeded.
Step: Server handshake succeeded.
Step: Master device accessible from companion.
Step: Added the servers 'sp7701b' and 'sp7501a' for cluster configuration.
Step: Server configuration initialization succeeded.
Step: Synchronizing Application Specific information from companion server
Step: Synchronizing Roles from companion server
Step: Synchronizing Login Roles from companion server
Step: Synchronizing Remote Logins from companion server
Step: Synchronizing Groups in sysusers from companion server
Step: Synchronizing Sysattributes from companion server
Step: Synchronizing server logins from companion server
Step: Synchronizing server-wide privs from companion server
Step: User information synchronization succeeded.
Step: Server configured in normal companion mode.
(return status = 0)
```

ASE companion server setup is completed.

1> sp_companion

```
2> go
Server 'sp7701b' is alive and cluster configured.
Server 'sp7701b' is configured for HA services.
Server 'sp7701b' is currently in 'Symmetric normal' mode.
(return status = 0)
```

Testing of SAP ASE on PowerHA

The team performed the following test scenarios, ASE server was successfully failed over to the other node using the PowerHA clustering service. The failover validation was done by running the `isql` client from a third node, and made sure that all the databases and devices were failed over successfully.

Scenario 1: Shut down ASE on one node

Use `isql` to shut down ASE.

Scenario 2: Kill the ASE process on one node

```
kill -9 <ASE process ID>
```

Scenario 3: Move the resource group to the other node

Use smitty to move the resource group from one node to the other node.

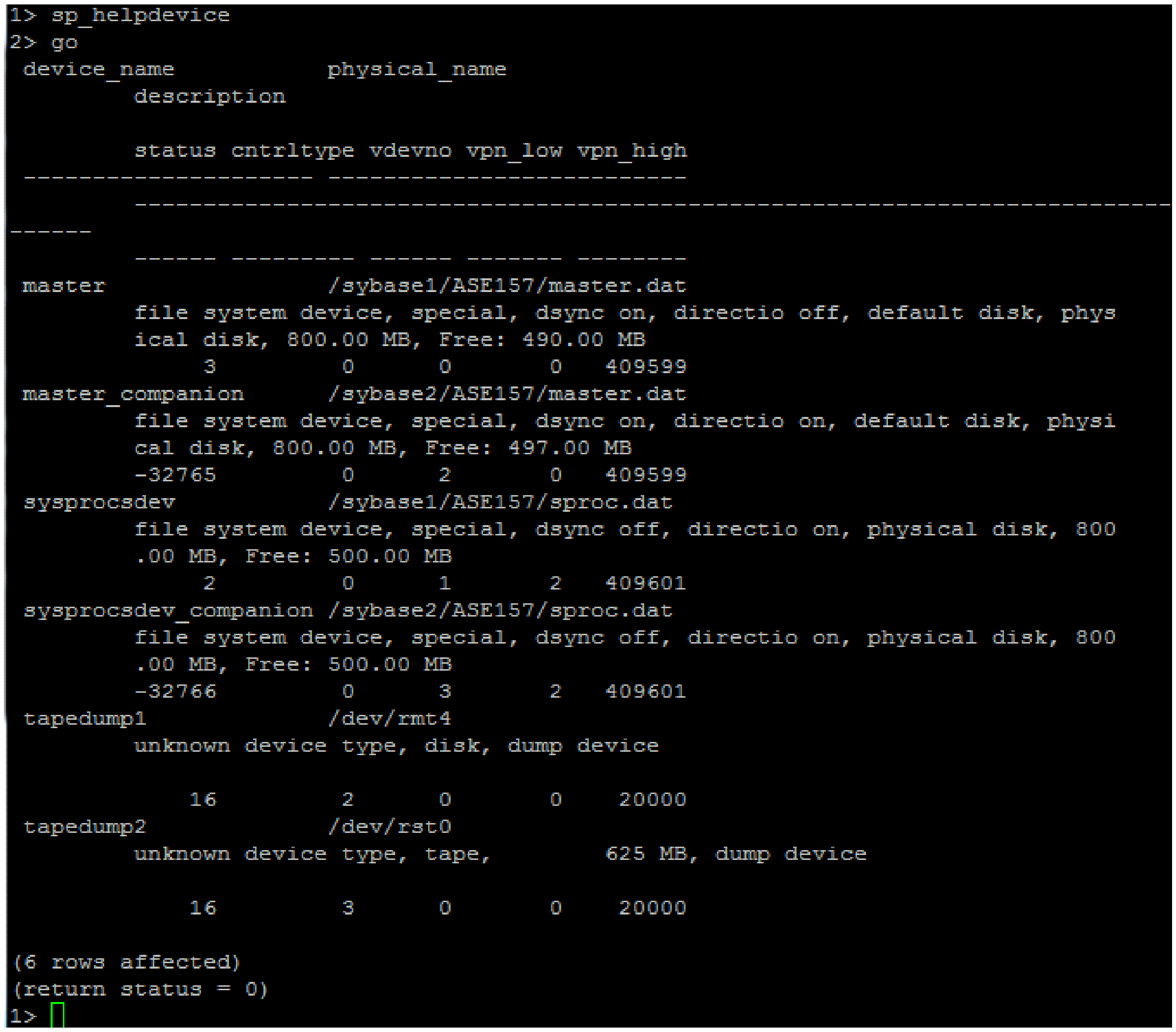
Scenario 4: Shut down the cluster service on one node

Use smitty to shut down the cluster service on one node.

Scenario 5: Shut down one node

Use the AIX shutdown command to shut down one node.

In this particular example, Node B ASE databases were successfully failed over to Node A, and were hosted by the ASE server as **master_companion** and **sysprocesdev_companion** databases.

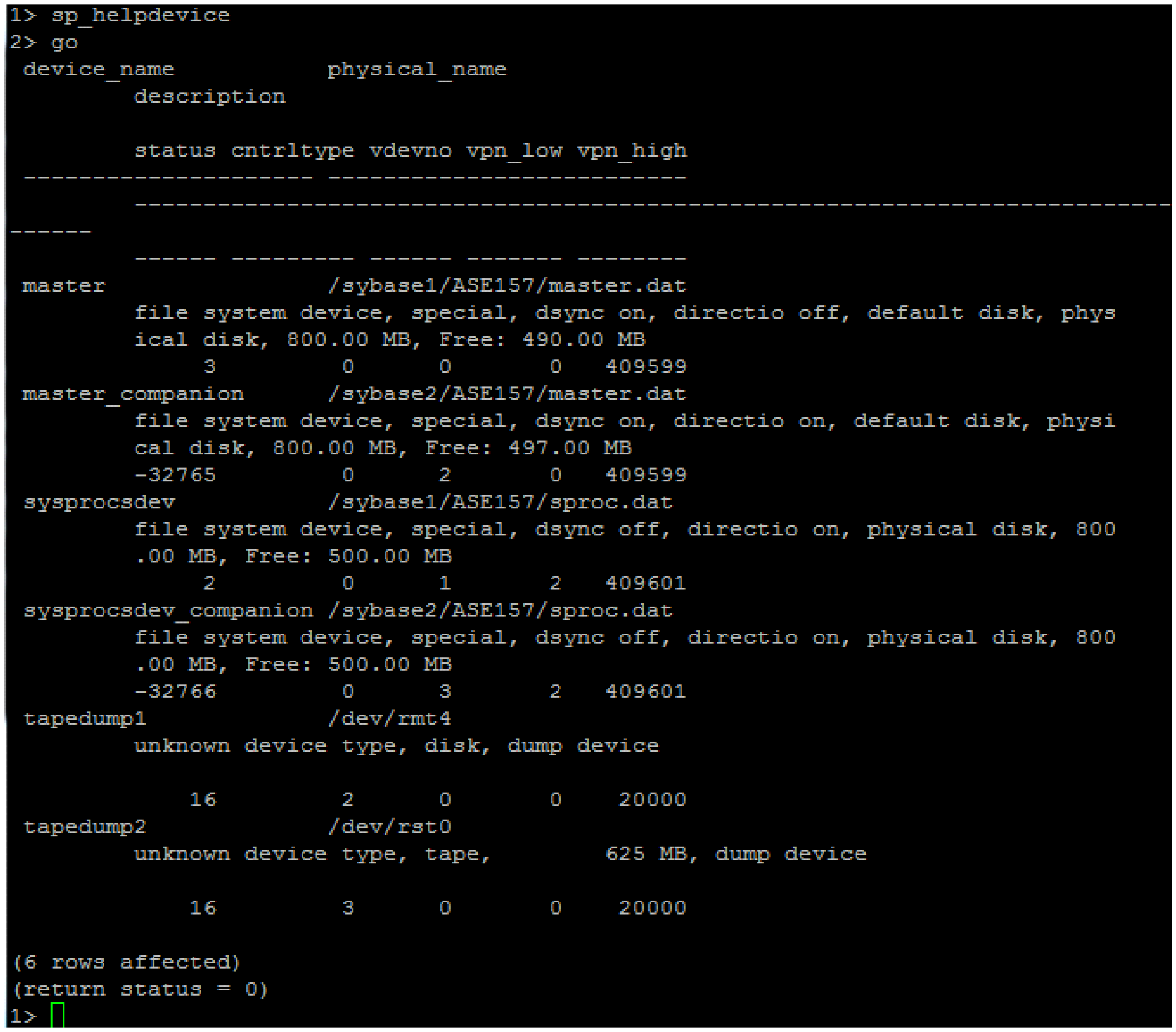


```
1> sp_helpdevice
2> go
 device_name           physical_name
         description

         status cntrltype vdevno vpn_low vpn_high
 ---------------------- --------------------------
         ------------------------------------------------------------------------
------
         ------ --------- ------ ------- --------
 master                /sybase1/ASE157/master.dat
         file system device, special, dsync on, directio off, default disk, phys
         ical disk, 800.00 MB, Free: 490.00 MB
             3         0      0       0   409599
 master_companion      /sybase2/ASE157/master.dat
         file system device, special, dsync on, directio on, default disk, physi
         cal disk, 800.00 MB, Free: 497.00 MB
         -32765        0      2       0   409599
 sysprocsdev           /sybase1/ASE157/sproc.dat
         file system device, special, dsync off, directio on, physical disk, 800
         .00 MB, Free: 500.00 MB
             2         0      1       2   409601
 sysprocsdev_companion /sybase2/ASE157/sproc.dat
         file system device, special, dsync off, directio on, physical disk, 800
         .00 MB, Free: 500.00 MB
         -32766        0      3       2   409601
 tapedump1             /dev/rmt4
         unknown device type, disk, dump device

            16         2      0       0    20000
 tapedump2             /dev/rst0
         unknown device type, tape,        625 MB, dump device

            16         3      0       0    20000

(6 rows affected)
(return status = 0)
1>
```

Figure 5: Device status post failover

In this particular example, Node B ASE devices were successfully failed over to Node A, and were hosted by the ASE server on node A.

Note: Node B ASE devices was installed in /sybase2/ASE157/master.dat and /sybase2/ASE157/sproc.dat on Node B.



```
1> sp_helpdevice
2> go
 device_name           physical_name
         description

         status cntrltype vdevno vpn_low vpn_high
 --------------------- --------------------------
         --------------------------------------------------------------------------
------
         ------ --------- ------ ------- --------
 master                /sybase1/ASE157/master.dat
         file system device, special, dsync on, directio off, default disk, phys
         ical disk, 800.00 MB, Free: 490.00 MB
              3         0      0       0   409599
 master_companion      /sybase2/ASE157/master.dat
         file system device, special, dsync on, directio on, default disk, physi
         cal disk, 800.00 MB, Free: 497.00 MB
         -32765         0      2       0   409599
 sysprocsdev           /sybase1/ASE157/sproc.dat
         file system device, special, dsync off, directio on, physical disk, 800
         .00 MB, Free: 500.00 MB
              2         0      1       2   409601
 sysprocsdev_companion /sybase2/ASE157/sproc.dat
         file system device, special, dsync off, directio on, physical disk, 800
         .00 MB, Free: 500.00 MB
         -32766         0      3       2   409601
 tapedump1             /dev/rmt4
         unknown device type, disk, dump device

             16         2      0       0    20000
 tapedump2             /dev/rst0
         unknown device type, tape,        625 MB, dump device

             16         3      0       0    20000

(6 rows affected)
(return status = 0)
1>
```

Figure 6: Device status post failover (continued)

Summary

Running SAP ASE on the AIX operating system in high availability provides maximum uptime for business-critical applications. .

Appendix A: Sample ASE HA script (ASE_HA.sh)

This section shows a sample ASE HA script that is available on ASE media, which was modified for this exercise using Companion server

For a sample ASE-HA script for stand-alone active-passive failover, refer to “Appendix B: Sample ASE HA script changes for PowerHA active-passive failover”.

Sample ASE HA script (ASE_HA.sh)

```
#!/usr/bin/sh
# ************************************************************************
# ***  Startup, Shutdown and Monitor Script for SYBASE ASE (Template)  ***
# ************************************************************************
# ******** Note: This file MUST be edited before it can be used. *********
# ************************************************************************
#
#       Note: To ensure high availability and security, the following
#       rules must be adhere to when using this script:
#       1. This script MUST be owned by 'root' and has read, write,
#          execute permission only for 'root'.
#       2. This script MUST be used in conjunction with an ASE 12.0 server
#          or later that is configured for HA failover.
#       3. No clients should be allowed to login directly to the ASE server
#          on the current node.
#
# ************************************************************************
#
# User-Configurable Variables:
#
# MONITOR_INTERVAL:
#       The interval of time in seconds, which this script should
#       wait between checks of the existence of the ASE dataserver.
#
# RECOVERY_TIMEOUT:
#       The maximum time to wait for an ASE recovery to complete (in
#       seconds) before determining that the server had failed to start.
#       Therefore, set it to a reasonable maximum amount of time that it
#       should take for all your databases to come online, considering the
#       worst case scenario when your machine is fully loaded.  This is
#       also used as the maximum time allowed to wait for failover and
#       failback to complete.
#       * NOTE: This value should always be less than the wait time of
#       HACMP before it goes into a config_too_long state, which by
#       default is 360 seconds.  If there is a possibility that your server
#       will take longer than this to boot up, you can reconfigure this
#       value by executing the following command on HACMP:
#               chssys -s clstrmgr -a "-u milliseconds_to_wait"
#       The same applies to SHUTDOWN_TIMEOUT.
#
# SHUTDOWN_TIMEOUT:
#       The maximum time to wait for ASE server to shutdown before resorting
#       to killing it.
#
# CONNECTION_TIMEOUT:
#        The maximum time allowed for a connection to be established, before
#        isql times out.  Default is 120 seconds.
#
# RESPONSE_TIMEOUT:
#       The maximum time allowed for a simple query to return.  This
#       is used to diagnose server hangs.  Usually, if isql fails to
#       connect in CONNECTION_TIMEOUT period, it will automatically timeout
#       and exit.  However, in the case where a connection was successful
#       but a query does not return, this variable will help the monitor
#       determine that a server is hung.
#       If this is not set, it will default to 999999.
#
# ASE_FAILOVER:
#       yes - Monitor the ASE for hung or dead processes, and stop HACMP
#           on this node to failover the devices to the secondary server.
#           If set to "yes", you must run "sp_companion configure" on the
#           server as well to keep the high availability consistent.  You
#           must also use "failover" as the stop script argument when
#           defining this application server within HACMP.
#       no - Do not bring down HACMP on this node even if ASE were
#           to fail.  Useful when one needs to bring down the ASE for
#           maintenance/reconfiguration.
#       NOTE:
#       ** This should only be set to "yes" if BOTH the servers are running
#          ASE version 12.0 or later.  Servers of previous versions should
#          have this set to "no".
#       ** For an ASYMMETRIC SECONDARY server script, it should always be "no".
#
# BASIC_FAILOVER:
#       yes - Revert to the basic node failover provided by HACMP if
#           the servers are not in companionship mode, i.e. reboot the
#           primary server fresh on the secondary node.  For this to take
#           effect, you must use "failover" as the stop script argument
#           when defining this application server within HACMP.
#       no - Do not revert to basic node failover.  Therefore, if the servers
#           the servers are not in companionship during a failover, simply
#           exit.
#
# RETRY:
#       Number of attempts to boot the primary ASE server on the local node
#       in the case of failed server, before ending this monitoring or
#       stopping the HACMP.  Set this to 0 if you do not want the
#       server to reboot automatically upon failure.
#
# SYBASE_ASE:
#       The directory name under $SYBASE where the ASE products are
#       installed, i.e. ASE-15_0.
#
# SYBASE_OCS:
#       The directory name under $SYBASE where the OCS products are
#       installed, i.e. OCS-15_0.  This directory's bin should contain
#       the binary file 'isql'.
#
# PRIM_SERVER:
#       The ASE server which this script is monitoring (i.e. if this is
#       a script for the secondary server, set this variable to be
#       the secondary server name.
#
# SEC_SERVER:
#       Secondary Server which is the companion of the Primary Server.
#       For an asymmetric companionship, this would be the secondary server
#       name, regardless of whether this script monitors the primary
#       or the secondary server.
#
# PRIM_HOST/SEC_HOST:
#       Hostname of the service interface for the primary and secondary
#       server, respectively.
#
# PRIM_SYBASE/SEC_SYBASE:
#       Where the environment variable "$SYBASE" should be set to in order
#       to bring up the primary server on the primary and secondary nodes,
#       respectively.
#       If using local device, this location must be the same on both nodes.
#       If using shared device, this location must be different on both nodes.
#
# PRIM_SYBASE_HOME/SEC_SYBASE_HOME:
#       The install directory of Sybase ASE on the primary and secondary
#       hosts, respectively (usually $SYBASE/$SYBASE_ASE).
```

```
#
# PRIM_ISQL/SEC_ISQL:
#       Complete path to the 'isql' binary in the primary and secondary
#       hosts, respectively.
#
# HA_LOGIN:
#       Login of user with 'sa_role' and 'ha_role'.  This has to be the
#       same on both primary and secondary server.
#
# HA_PWD:
#       Password of ${HA_LOGIN}.  This has to be the same on both primary
#       and secondary server.
#
# CIPHER:
#       Encrypted Login and Password of user with 'sa_role' and 'ha_role'.
#       This has to be the same on both primary and secondary server. It
#       is the encrypted string you get from the haisql utility (located
#       in $SYBASE/$SYBASE_ASE/bin). CIPHER and (HA_LOGIN/HA_PWD) are
#       mutually exclusive. If you set CIPHER then the script uses the
#       encrypted login and password. Do not set HA_LOGIN and  HA_PWD
#       if CIPHER is set.
#
# PRIM_RUNSCRIPT
#       RUN_server script created by ASE installation to bring up the
#       primary server.
#
# PRIM_CONSOLE_LOG/SEC_CONSOLE_LOG:
#       Errorlog for this particular primary or secondary server session.
#       It can be any file in a filesystem with sufficient filespace
#       and writeable by root.  Default is in ${*_SYBASE_HOME}/install.
#
# Usage:
# * This script is to be entered into the HACMP for AIX 4.2.2 Resource
#   Configuration as part of the ASE Application Server Start/Stop script.
#
# * It will be called by HACMP event handlers 'start_server' and 'stop_server'.
#
# * Its output will be written to /tmp/hacmp.out along with the other HACMP
#   event handler output.
#
# * Its entry into the HACMP Resource Definition should be as follows:
# - For an Asymmetric Configuration Environment:
#       Server1(primary companion):
#               Start Script:  <script_name> monitor
#               Stop Script:   <script_name> failover
#
#       Server2(secondary companion):
#               Start Script:  <script_name> monitor
#               Stop Script:   <script_name> stop
#
# - For a Symmetric Configuration Environment:
#       Server1(primary companion):
#               Start Script:  <script_name> monitor
#               Stop Script:   <script_name> failover
#
#       Server2(primary companion):
#               Start Script:  <script_name> monitor
#               Stop Script:   <script_name> failover
#
################################################################################

PROG=${0}

################################################################################
# Set the variables below according to your cluster companionship setup
################################################################################
#
# Monitoring variables
#
```

```
MONITOR_INTERVAL=30
RECOVERY_TIMEOUT=300
SHUTDOWN_TIMEOUT=30
CONNECTION_TIMEOUT=120
RESPONSE_TIMEOUT=
RETRY=1
ASE_FAILOVER=yes
BASIC_FAILOVER=yes

#
# General environment settings
# Note: Customers should __FILL_IN__ the values for SYBASE_ASE and SYBASE_OCS
#       for their environment if the two varables are not default values.
#
export SYBASE_ASE=ASE-15_0
export SYBASE_OCS=OCS-15_0
export PATH=/sbin:/usr/sbin:/usr/bin
HA_LOGIN=__FILL_IN__
HA_PWD=__FILL_IN__
CIPHER=__FILL_IN__

#
# Variables for Primary Companion
#
PRIM_SERVER=__FILL_IN__
PRIM_HOST=__FILL_IN__
PRIM_SYBASE=__FILL_IN__
PRIM_SYBASE_HOME=${PRIM_SYBASE}/${SYBASE_ASE}
PRIM_ISQL=${PRIM_SYBASE}/${SYBASE_ASE}/bin/haisql
PRIM_CONSOLE_LOG=${PRIM_SYBASE_HOME}/install/${PRIM_SERVER}.cs_log
PRIM_RUNSCRIPT=${PRIM_SYBASE_HOME}/install/RUN_${PRIM_SERVER}

#
# Variables for Secondary Companion
#
SEC_SERVER=__FILL_IN__
SEC_HOST=__FILL_IN__
SEC_SYBASE=__FILL_IN__
SEC_SYBASE_HOME=${SEC_SYBASE}/${SYBASE_ASE}
SEC_ISQL=${SEC_SYBASE}/${SYBASE_ASE}/bin/haisql
SEC_CONSOLE_LOG=${SEC_SYBASE_HOME}/install/${SEC_SERVER}.cs_log

################################################################################
# End of user-configurable variables
################################################################################

HOST=`hostname`
USER=`whoami`          # This user needs to have execute permission on clstop
SYBUSER=sybase

if [ "${HOST}" = "${PRIM_HOST}" ]
then
    SYBASE=${PRIM_SYBASE}
else
    SYBASE=${SEC_SYBASE}
fi
export SYBASE

if [ "${RESPONSE_TIMEOUT}" = "" ]
then
    RESPONSE_TIMEOUT=999999
fi

# We should set LIBPATH so haisql can load properly.
# It depends on libsbgse2.so (encryption related libs)
if test -f $SYBASE/SYBASE.sh; then
```

```
        . $SYBASE/SYBASE.sh
else
        export
LIBPATH="$SYBASE/$SYBASE_ASE/lib:$SYBASE/$SYBASE_OCS/lib:$SYBASE/$SYBASE_OCS/lib3p64:$SYBASE/$SYBA
SE_OCS/lib3p"
fi

################################################################################
#
# Function start_ASE_as_sybase_user
#       Starts ASE as sybase user using su command; sets
#       required env or sources $SYBASE/SYBASE.sh file
#       inside the ksh run via su.
#
################################################################################

function start_ASE_as_sybase_user
{
        # When passing commands to ksh using <<, escape all
        # special characters such as $, `, "  etc., Otherwise,
        # the meaning of the special chars will be applied
        # in the current shell, not the ksh which is invoked
        # after su'ing as sybase user. We don't escape the
        # special chars where we intend to use the current
        # shell to interpret these chars. Note that EOF must
        # start on a different line.
        #
        # Also, we don't use stderr/stdout redirection during
        # su, becoz file desc 1/2 is not identified by some
        # shells like csh (default shell for SYBUSER may vary
        # by environments) and this may fail whole operation.

        su ${SYBUSER} -c ksh << EOF
                # set required SYBASE environment. Note that PATH and
                # LIBPATH are NOT propagated by su, so set them again.
                # Other env variables will be received from parent.
                if test -f $SYBASE/SYBASE.sh; then
                        . $SYBASE/SYBASE.sh
                else
                        export
LIBPATH="$SYBASE/$SYBASE_ASE/lib:$SYBASE/$SYBASE_OCS/lib:$SYBASE/$SYBASE_OCS/lib3p64:$SYBASE/$SYBA
SE_OCS/lib3p"
                fi

                . ${PRIM_RUNSCRIPT} > ${PRIM_CONSOLE_LOG} 2>&1 &
EOF
}

################################################################################
# Function: sybase_run_cmds
#
# If executed on the primary node, this function is used to start the primary
# dataserver instance.
# If executed on the secondary node, it is used to execute a 'DBCC TAKEOVER'
# during an ASE server failover.  However, if a takeover is not applicable,
# it will start the primary dataserver on the secondary node if BASIC_FAILOVER
# is set to "yes".
#
################################################################################

function sybase_run_cmds
{
#   trap "" SIGTERM

    print "$(date +'%b %e %T') SYBASE HA MONITOR: Start 'sybase_run_cmds'"
    if [[ "${PRIM_HOST}" = "${HOST}" ]]
    then
```

```
        #
        # Boot up the dataserver on primary node
        #
        rm -f ${PRIM_CONSOLE_LOG}
        print "$(date +'%b %e %T') SYBASE HA MONITOR: Starting '${PRIM_SERVER}' as ${SYBUSER}."

        start_ASE_as_sybase_user

        #
        # Monitor every 30 seconds if that the server has recovered, until
        # RECOVERY_TIMEOUT.
        #
        print "Please wait..."

        t=0
        while [[ $t -lt ${RECOVERY_TIMEOUT} ]]
        do
            sleep 3
            grep -s "Recovery complete." ${PRIM_CONSOLE_LOG} >> /dev/null
            if [[ $? != 0 ]]
            then
                t=`expr $t + 3`
                 print ".\c"
            else
                print "\n$(date +'%b %e %T') SYBASE HA MONITOR: ASE server '${PRIM_SERVER}' started
successfully."
                break
            fi
        done

        #
        # Check one last time if the server is up.
        #
        grep -s "Recovery complete." ${PRIM_CONSOLE_LOG} >> /dev/null
        if [[ $? != 0 ]]
        then
             print "$(date +'%b %e %T') SYBASE HA MONITOR: ***ERROR: ASE server '${PRIM_SERVER}'
failed to start.  Please see error in ${PRIM_CONSOLE_LOG}, correct the problem, and restart by
stopping and starting HACMP on this node."
        fi
    else

        #
        # If the servers are in a companionship, we execute dbcc takeover
        # from the secondary server to takeover the primary databases.
        # If the servers are not in a companionship and BASIC_FAILOVER is
        # set, we reboot the primary server on the secondary node.
        #
        CMPSTATE=`${SEC_ISQL} ${LOGIN_STRING} -S${SEC_SERVER} -I${SEC_SYBASE}/interfaces -
CGET_CMPSTATE `

        print "CMPSTATE=$CMPSTATE"

        if [[ ${CMPSTATE} = "" ]]
        then
            print "SYBASE HA MONITOR: ASE server '${SEC_SERVER}' is either down or inaccessible."
            CMPSTATE=0
        fi

        case ${CMPSTATE} in
        0|-2)
            # This is the case of a single server

            print "$(date +'%b %e %T') SYBASE HA MONITOR: Using basic node failover if
BASIC_FAILOVER is set."
            if [[ "${BASIC_FAILOVER}" = "yes" ]]
            then
                print "$(date +'%b %e %T') SYBASE HA MONITOR: BASIC_FAILOVER is set.  Starting ASE
server '${PRIM_SERVER}' on ${HOST}."
                export SYBASE="${PRIM_SYBASE}"
               start_ASE_as_sybase_user
               print "Please wait..."

               t=0
               while [[ $t -lt ${RECOVERY_TIMEOUT} ]]
               do
                   sleep 3
                   grep -s "Recovery complete." ${PRIM_CONSOLE_LOG} >> /dev/null
                   if [[ $? != 0 ]]
                   then
                       t=`expr $t + 3`
                       print ".\c"
                   else
                       print "\n$(date +'%b %e %T') SYBASE HA MONITOR: ASE server '${PRIM_SERVER}'
started successfully."
                        break
                   fi
                done

               grep -s "Recovery complete." ${PRIM_CONSOLE_LOG} >> /dev/null
               if [[ $? != 0 ]]
               then
                   print "$(date +'%b %e %T') SYBASE HA MONITOR: ***ERROR: ASE server
'${PRIM_SERVER}' failed to come up on '${HOST}'."
               fi

            else

                print "$(date +'%b %e %T') SYBASE HA MONITOR: BASIC_FAILOVER is not set. Do
nothing."

            fi
            ;;

        2|11)
            # This is the case of a companion server taking over

            print "$(date +'%b %e %T') SYBASE HA MONITOR: '${SEC_SERVER}' is a normal companion of
'${PRIM_SERVER}'."
            print "$(date +'%b %e %T') SYBASE HA MONITOR: Executing 'DBCC TAKEOVER' on
'${SEC_SERVER}'."

            ${SEC_ISQL} ${LOGIN_STRING} -S${SEC_SERVER} -I${SEC_SYBASE}/interfaces -CDBCC_TAKEOVER
&

            print "Please wait..."

            t=0
            while [[ $t -le ${RECOVERY_TIMEOUT} ]]
            do
               tail ${SEC_CONSOLE_LOG} | grep -s "Failover of companion server is completed
successfully." >> /dev/null
               if [[ $? = 0 ]]
               then
                   print "\n$(date +'%b %e %T') SYBASE HA MONITOR: DBCC TAKEOVER of
'${PRIM_SERVER}' was successful."
                   break
                else
                   sleep 5
                   t=`expr $t + 5`
                   print ".\c"
                fi
            done

            if [[ $t -gt ${RECOVERY_TIMEOUT} ]]
            then
                print "$(date +'%b %e %T') SYBASE HA MONITOR: ***ERROR: Dbcc takeover may have 
failed.  Please check the errorlog of ${SEC_SERVER} and manually execute 'dbcc takeover' from 
${SEC_SERVER} if necessary."
            fi
            ;;

        *)
            # For other modes - simply print a message

            print "$(date +'%b %e %T') SYBASE HA MONITOR: '${SEC_SERVER}' is neither a single 
server nor a normal companion of '${PRIM_SERVER}', therefore do nothing."
            ;;

        esac
    fi
    print "$(date +'%b %e %T') SYBASE HA MONITOR: End 'sybase_run_cmds'."
}

################################################################################
# Function: sybase_stop_cmds
#
# Shut down the primary server and kill any hung processes that is still
# lingering around.
#
################################################################################

function sybase_stop_cmds
{
    print "$(date +'%b %e %T') SYBASE HA MONITOR: Start 'sybase_stop_cmds'."

    #
    # Just in case things are hung, start a process that will wait for the
    # SHUTDOWN_TIMEOUT period, then kill any remaining processes.  We'll
    # need to monitor this pid so we can terminate it later if it is not
    # needed.
    #
    ${PROG} kill &
    KILLJOB=$!

    #
    # If the server is up and running, shut it down first.
    #
    ${PRIM_ISQL} ${LOGIN_STRING} -S${PRIM_SERVER} -I${PRIM_SYBASE}/interfaces -CSHUTDOWN "with 
nowait" &

    print "Please wait..."

    t=0
    while [[ $t -lt ${SHUTDOWN_TIMEOUT} ]]
    do
        tail ${PRIM_CONSOLE_LOG} | grep "ueshutdown: exiting" >> /dev/null
        if [[ $? = 0 ]]
        then
             print "\n$(date +'%b %e %T') SYBASE HA MONITOR: ASE server '${PRIM_SERVER}' shutdown 
successfully."
            break;
        else
            print ".\c"
            sleep 2
            t=`expr $t+2`
        fi
    done

    if [[ $t -ge ${SHUTDOWN_TIMEOUT} ]]
    then
        print "$(date +'%b %e %T') SYBASE HA MONITOR: ASE server '${PRIM_SERVER}' failed to
shutdown.  Server is either down or unreachable."
    fi

    #
    # Get the pid of engine 0 from the errorlog and if it is no longer running,
    # we can stop the kill process.
    #
        set -A KERNEL_MODE `sed -n -e 's/^.*Adaptive Server is using the \([a-z]*\) kernel
mode./\1/p' ${PRIM_CONSOLE_LOG}`

        if [ "${KERNEL_MODE}" = "threaded" ]
        then
                engine_0=`sed -n -e 's/^.*Adaptive Server is running as process id \([0-
9]*\)./\1/p' ${PRIM_CONSOLE_LOG}`
        else
                engine_0=`sed -n -e '/engine 0/s/^.*os pid \([0-9]*\).*$/\1/p' ${PRIM_CONSOLE_LOG}`
        fi

    running=`ps -fu ${SYBUSER}|grep " ${engine_0} "|grep -v grep | awk '{print $2}'`
    if [[ "${running}" = "" ]]
    then
        print "$(date +'%b %e %T') SYBASE HA MONITOR: Killing the kill_hung_processes script."
         kill -9 $KILLJOB
    fi
    print "$(date +'%b %e %T') SYBASE HA MONITOR: End 'sybase_stop_cmds'."
}

###############################################################################
# Function: terminate
#
# Called when a SIGTERM is trapped.
#
###############################################################################

function terminate
{
     print "$(date +'%b %e %T') SYBASE HA MONITOR: This monitor script has been signaled to
terminate."
     exit
}

###############################################################################
# Function: kill_hung_processes
#
# In case the dataserver hangs and cannot be shut down,  we spawn a
# background process simultaneously to kill each of the engine processes
# after a timeout period.  This guarantees a clean stop_server attempt.
#
###############################################################################

function kill_hung_processes
{
    #
    # Wait for awhile to allow processes to clean up before killing all
    # dataserver processes.
    #
    print "$(date +'%b %e %T') SYBASE HA MONITOR: Start 'kill_hung_processes'."

    print "Please wait..."
    sleep ${SHUTDOWN_TIMEOUT}

        set -A KERNEL_MODE `sed -n -e 's/^.*Adaptive Server is using the \([a-z]*\) kernel
mode./\1/p' ${PRIM_CONSOLE_LOG}`

        if [ "${KERNEL_MODE}" = "threaded" ]
        then
                set -A engine_n `sed -n -e 's/^.*Adaptive Server is running as process id \([0-
9]*\)./\1/p' ${PRIM_CONSOLE_LOG}`
        else
                set -A engine_n `sed -n -e '/engine /s/^.*os pid \([0-9]*\).*online$/\1/p'
${PRIM_CONSOLE_LOG}`
        fi

    for id in ${engine_n[@]}
    do
        kill -15 ${id}
    done

    print "$(date +'%b %e %T') SYBASE HA MONITOR: End 'kill_hung_processes'."
}

################################################################################
# Function: stop_hacmp_services
#
# Shutdown the HACMP services on the primary node so that all the
# primary resources could fail over to the secondary node.
#
################################################################################
function stop_hacmp_services
{
    print "$(date +'%b %e %T') SYBASE HA MONITOR: Start 'stop_hacmp_services'."
    if [[ "${PRIM_HOST}" = "${HOST}" ]]
    then
        if [[ $retry = 0 ]]
        then
            if [[ "${ASE_FAILOVER}" = "yes" ]]
            then
                #
                 # All retries have been exhausted, so bring down HACMP with
                # Shutdown mode="takeover" to allow the second node to acquire
                # the devices of this application server.
                #
                print "$(date +'%b %e %T') SYBASE HA MONITOR: Shutting down HACMP services to allow
devices to failover to '${SEC_HOST}'."
                /usr/sbin/cluster/utilities/clstop -y -N -gr
            else
                print "$(date +'%b %e %T') SYBASE HA MONITOR: ASE_FAILOVER not set -- no action
taken."
            fi
        else
            print "$(date +'%b %e %T') SYBASE HA MONITOR: retry is not zero -- no action taken."
        fi
    fi
    print "$(date +'%b %e %T') SYBASE HA MONITOR: End 'stop_hacmp_services'."
}

################################################################################
# Function: failover
#
# The function is called by the HACMP stop_server event handler.
# If executed on primary, it will bring down the server and clean up all
# remaining processes.
# If executed on secondary, it will issue a 'prepare_failback' from
# the secondary companion server or, if the server is not in companionship
# AND BASIC_FAILOVER is set to "yes", shut the server down on the current node.
```

```
#
################################################################################
function failover
{
    print "$(date +'%b %e %T') SYBASE HA MONITOR: Start 'failover'."
    if [[ "${PRIM_HOST}" = "${HOST}" ]]
    then
        #
        # First try to determine if this failover was triggered by our own
        # monitoring script or by HACMP shutdown from other reasons.  If
        # the latter, we want to kill the monitoring process so we don't
        # get into a loop of monitoring our own failover steps.
        #
        ps -fu ${USER} | grep "${PROG} monitor" | grep -v grep
        MONPID=`ps -fu ${USER} | grep "${PROG} monitor" | grep -v grep | awk '{print $2}'`
        if [[ "$MONPID" != "" ]]
        then
            print "$(date +'%b %e %T') SYBASE HA MONITOR: Killing the monitoring process
'${MONPID}' first."
            kill -15 $MONPID
        else
            print "$(date +'%b %e %T') SYBASE HA MONITOR: Monitoring script is not running."
        fi

        #
        # We are failing over to the secondary node.  Shutdown the Primary
        # Server (if it is not already down) by executing sybase_stop_cmds.
        #
         sybase_stop_cmds

    else

        #
        # We are failing back to primary node.
        # First determine if we are in a failedover mode.  If we are,
        # run 'sp_companion prepare_failback' on the secondary to release
        # all the Primary Server's resources.  If we are in a single server
        # mode, or if the secondary server is inaccesible or not HA-enabled,
        # simply shut down the primary server on the secondary node.
        #

        CMPSTATE=`${SEC_ISQL} ${LOGIN_STRING} -S${SEC_SERVER} -I${SEC_SYBASE}/interfaces -
CGET_CMPSTATE `

        print "CMPSTATE=$CMPSTATE"

        if [[ ${CMPSTATE} = "" ]]
        then
            print "SYBASE HA MONITOR: Server '${SEC_SERVER}' is either down or inaccessible."
            CMPSTATE=0
        fi

         case ${CMPSTATE} in
         0|-2)
            # This is the case of a single server

            print "SYBASE HA MONITOR: Using basic node failover if BASIC_FAILOVER is set."
             if [[ "${BASIC_FAILOVER}" = "yes" ]]
             then
                print "SYBASE HA MONITOR: BASIC_FAILOVER is set.  Shutdown '${PRIM_SERVER}' on
${HOST}."
                 sybase_stop_cmds
            else
                print "SYBASE HA MONITOR: BASIC_FAILOVER is not set.  Therefore do nothing."
             fi
             ;;
        4|12)
            # This is the case of a companion server failing back

            print "SYBASE HA MONITOR: '${SEC_SERVER}' is a failed over companion for
'${PRIM_SERVER}'."
            print "SYBASE HA MONITOR: Failback from Secondary to Primary Server."
            ${SEC_ISQL} ${LOGIN_STRING} -S${SEC_SERVER} -I${SEC_SYBASE}/interfaces -
CPREPARE_FAILBACK ${PRIM_SERVER} &

            print "Please wait..."

            t=0
            while [[ $t -le ${RECOVERY_TIMEOUT} ]]
            do
               tail ${SEC_CONSOLE_LOG} | grep -s "Failback completed." >> /dev/null
               if [[ $? = 0 ]]
               then
                   print "\n$(date +'%b %e %T') SYBASE HA MONITOR: Prepare_failback was
successful."
                   break
               else
                   print ".\c"
                   sleep 5
                   t=`expr $t + 5`
               fi
            done

            if [[ $t -gt ${RECOVERY_TIMEOUT} ]]
            then
                   print "$(date +'%b %e %T') SYBASE HA MONITOR: ***ERROR: Prepare_failback
failed.  Please check ${SEC_CONSOLE_LOG} and refer to the documentation on how to proceed."
            fi
            ;;

        *)
            # For other modes - simply print a message

            print "SYBASE HA MONITOR: '${SEC_SERVER}' is neither a single server nor a failed over
companion for '${PRIM_SERVER}', therefore do nothing."
            ;;

        esac
    fi
    print "$(date +'%b %e %T') SYBASE HA MONITOR: End 'failover'."
}

################################################################################
# Function: monitor_processes
#
# Monitor the primary server and trigger a failover event if the server is
# deemed to be non-responsive or dead.
#
################################################################################

function monitor_processes
{
    if [[ "${PRIM_HOST}" != "${HOST}" ]]
    then
        #
        # Noopt for Secondary server
        #
        return
    fi
    print "$(date +'%b %e %T') SYBASE HA MONITOR: Start 'monitor_processes'."

    stop_hacmp=0

        set -A KERNEL_MODE `sed -n -e 's/^.*Adaptive Server is using the \([a-z]*\) kernel
mode./\1/p' ${PRIM_CONSOLE_LOG}`

        if [ "${KERNEL_MODE}" = "threaded" ]
        then
                engine_0=`sed -n -e 's/^.*Adaptive Server is running as process id \([0-
9]*\)./\1/p' ${PRIM_CONSOLE_LOG}`
        else
                engine_0=`sed -n -e '/engine 0/s/^.*os pid \([0-9]*\).*$/\1/p' ${PRIM_CONSOLE_LOG}`
        fi

    if [[ "${engine_0}" = "" ]]
    then
        stop_hacmp=1
    fi

    if [[ ${stop_hacmp} = 1 ]]
    then
        print "$(date +'%b %e %T') SYBASE HA MONITOR: ASE server ${PRIM_SERVER} has failed to
start."
        set -m
        stop_hacmp_services # Reboot the server/stop the HACMP services
        set +m
    else
        print "$(date +'%b %e %T') SYBASE HA MONITOR: ASE Server '${PRIM_SERVER}' is now running
and being monitored.  Pid = ${engine_0}."
        retry=${RETRY}
        monitor=1
        timeout=0
        sleep ${MONITOR_INTERVAL}
        while [[ $monitor = 1 ]]
        do
            kill -s 0 ${engine_0} > /dev/null
            if [[ $? != 0 ]]
            then
                print "$(date +'%b %e %T') SYBASE HA MONITOR: Process pid ${engine_0} has died."
                monitor=0
            else
                # Option -d is used to print the diagnostic messages. In some
                # cases logic below does not work as expected. It is because
                # we write  some diagnostics messages when -d option is used,
                # later we check for the file size > 0 and grep for a specific
                # string to take further action.
                ${PRIM_ISQL} ${LOGIN_STRING} -S${PRIM_SERVER} -I${PRIM_SYBASE}/interfaces -
l${CONNECTION_TIMEOUT} -d -o/tmp/ase_poll.$$ -CCHECK_ALIVE &
                # In case select 1 hangs, we want to time out after awhile.
                s=0
                while [[ $s -lt ${RESPONSE_TIMEOUT} ]]
                do
                    if [[ -s /tmp/ase_poll.$$ ]]
                    then
                        grep -s "Read from the server has timed out" /tmp/ase_poll.$$ >> /dev/null
                        if [[ $? = 0 ]]
                        then
                            # If we are getting server timeout error 3 times in
                            # a row, it is safe to assume that server is hung
                            timeout=`expr $timeout + 1`
                            if [[ $timeout = 3 ]]
                            then
                                print "$(date +'%b %e %T') SYBASE HA MONITOR: isql connection
timeout.  ASE server '${PRIM_SERVER}' is hung."
                                monitor=0
                            else
```

```
                              print "$(date +'%b %e %T') SYBASE HA MONITOR: isql connection 
timeout.  Will retry after ${MONITOR_INTERVAL} seconds."
                              sleep ${MONITOR_INTERVAL}
                           fi
                       else
                           # If we did not get timeout error and select
                           # returns, assume that server is not hung, so
                           # reset timeout.
                           timeout=0
                       fi
                       break
                   else
                       sleep ${MONITOR_INTERVAL}
                       s=`expr $s + $MONITOR_INTERVAL`
                   fi
               done

               rm -f /tmp/ase_poll.$$

               if [ $s -ge ${RESPONSE_TIMEOUT} ]
               then
                   print "$(date +'%b %e %T') SYBASE HA MONITOR: '${PRIM_SERVER}' is hung.  Server 
did not return from a query."
                   monitor=0
               fi
           fi
       done

       if [[ $monitor = 0 ]]
       then
           set -m
           sybase_stop_cmds   # Stop/kill the hanging server.
           stop_hacmp_services # Reboot server/Stop the HACMP services
           set +m
       fi
   fi
   print "$(date +'%b %e %T') SYBASE HA MONITOR: End 'monitor_processes'."
}

################################################################################
# FUNCTION STARTUP SECTION.
#
# Test to see if we are being called to run the application, or stop the
# application.
#
################################################################################

trap terminate SIGTERM

print "$(date +'%b %e %T') SYBASE HA MONITOR: *** START with argument '${1}' ***"
LOGIN_STRING=""
if [[ ${CIPHER} = "__FILL_IN__" ]]
then
       LOGIN_STRING="-U${HA_LOGIN} -P${HA_PWD}"
else
       LOGIN_STRING="-E${CIPHER}"
fi

case $1 in

    fault)
        stop_hacmp_services
    ;;

    kill)
        kill_hung_processes
    ;;

    start)
        print "$(date +'%b %e %T') SYBASE HA MONITOR: Starting SYBASE ASE '${PRIM_SERVER}'."
        sybase_run_cmds
    ;;

    stop)
        print "$(date +'%b %e %T') SYBASE HA MONITOR: Stopping SYBASE ASE '${PRIM_SERVER}'."
        sybase_stop_cmds
    ;;

    monitor)
       retry=${RETRY}
       sybase_run_cmds
        monitor_processes
       if [[ "${PRIM_HOST}" = "${HOST}" ]]
       then
           while [[ ${retry} -gt 0 ]]
           do
               #
               # Try to reboot server on the same node instead of failing over
               #
               print "$(date +'%b %e %T') SYBASE HA MONITOR: Try to reboot ASE Primary Server on
current node ${retry} more time(s)."
               retry=`expr $retry - 1`
               #
               # First kill off any remaining processes, and restart the
               # monitoring process.
               #
               kill_hung_processes
               sybase_run_cmds
               monitor_processes
           done
       fi
    ;;

    failover)
       failover
    ;;

    *)
        print "Usage: ${SYBHA_MON} [ stop | start | monitor | failover ]"
    ;;
esac

print "$(date +'%b %e %T') SYBASE HA MONITOR: *** END '$1' ***"

exit 0
```

Appendix B: Sample ASE HA script changes for PowerHA active-passive failover

ASE_HA.sh

<<Contributed by Peter Barnett>>

There are Parameters that need to be configured for PowerHA active-passive failover. This section shows the key changes (in bold font) that need to be done to the ASE_HA.sh script.

```
###############################################################################
# Set the variables below according to your cluster companionship setup
###############################################################################
#
# Monitoring variables
#
MONITOR_INTERVAL=30
RECOVERY_TIMEOUT=300
SHUTDOWN_TIMEOUT=30
CONNECTION_TIMEOUT=120
RESPONSE_TIMEOUT=60
RETRY=1
ASE_FAILOVER=no
BASIC_FAILOVER=yes   #these choices specify that this is a simple active-passive
failover with no "companion server"

#
# General environment settings
# Note: Customers should __FILL_IN__ the values for SYBASE_ASE and SYBASE_OCS
#       for their environment if the two varables are not default values.
#
export SYBASE_ASE=ASE-15_0
export SYBASE_OCS=OCS-15_0
export PATH=/sbin:/usr/sbin:/usr/bin
HA_LOGIN=__FILL_IN__
HA_PWD=__FILL_IN__
CIPHER=

#
# Variables for Primary Companion
#
```

```
PRIM_SERVER=__FILL_IN__
PRIM_SERVER_BS=__FILL_IN__   #added by PB 6/13 to manage backup server
PRIM_HOST=__FILL_IN__
PRIM_SYBASE=/sybase
PRIM_SYBASE_HOME=${PRIM_SYBASE}/${SYBASE_ASE}
PRIM_ISQL=${PRIM_SYBASE}/${SYBASE_ASE}/bin/haisql
PRIM_CONSOLE_LOG=${PRIM_SYBASE_HOME}/install/${PRIM_SERVER}.cs_log
PRIM_CONSOLE_LOG_BS=${PRIM_SYBASE_HOME}/install/${PRIM_SERVER_BS}.cs_log
PRIM_RUNSCRIPT=${PRIM_SYBASE_HOME}/install/RUN_${PRIM_SERVER}
PRIM_RUNSCRIPT_BS=${PRIM_SYBASE_HOME}/install/RUN_${PRIM_SERVER_BS}  #added by PB 6/13
to manage backup server
#
# Variables for Secondary Companion
#
SEC_SERVER=
SEC_HOST=__FILL_IN__  #this may not have been required
SEC_SYBASE=
SEC_SYBASE_HOME=
SEC_ISQL=
SEC_CONSOLE_LOG=


      . ${PRIM_RUNSCRIPT} > ${PRIM_CONSOLE_LOG} 2>&1 &
      . ${PRIM_RUNSCRIPT_BS} > ${PRIM_CONSOLE_LOG_BS} 2>&1 &   #added by PB to start
backup server
```

PowerHA resource changes

A process monitor can and must be implemented for the SAP ASE *backup server* as well, along these lines.

Example:

```
Policies for sybase_rg resource group: (essentially all defaults)
 Resource Group Name                                   sybase_rg
 Participating Nodes (Default Node Priority)           atssg54 atssg55

 Startup Policy                                        Online On Home Node O>
 Fallover Policy                                       Fallover To Next Prio>
 Fallback Policy                                       Fallback To Higher Pr>
 Fallback Timer Policy (empty is immediate)           []                     +

 Service IP Labels/Addresses                          [atssg56]              +
 Application Controllers                            [fcsybase1_ctrl]       +

 Volume Groups                                      [sybasevg ]            +
 Use forced varyon of volume groups, if necessary    false                 +
 Automatically Import Volume Groups                  false                 +

 Filesystems (empty is ALL for VGs specified)       [ ]                    +
 Filesystems Consistency Check                       fsck                  +
 Filesystems Recovery Method                         sequential            +
 Filesystems mounted before IP configured            false                 +

 Filesystems/Directories to Export (NFSv2/3)        []                     +
 Filesystems/Directories to NFS Mount               []
 Network For NFS Mount                              []                     +

 Tape Resources                                     []                     +
 Raw Disk PVIDs                                     []                     +
 Raw Disk UUIDs/hdisks                              []                     +
 Disk Error Management?                              no                    +

 Primary Workload Manager Class                     []                     +
 Secondary Workload Manager Class                   []                     +

 Miscellaneous Data                                 []
 WPAR Name                                          []                     +
 User Defined Resources                             [ ]                    +
```

Application controller scripts

```
                                                       [Entry Fields]
 Application Controller Name                         fcsybase1_ctrl
 New Name                                           [fcsybase1_ctrl]
```
` Start Script                                       [/home/sybase/ASE_HA.sh monitor>`
`#just a name, does not actually monitor`
` Stop Script                                        [/home/sybase/ASE_HA.sh failover >`
` Application Monitor Name(s)                         fcsybase1_dataserver   +`
```
 Application startup mode                           [background]           +
```

Process monitor

```
                                                        [Entry Fields]
* Monitor Name                                        fcsybase1_dataserver
  Application Controller(s) to Monitor                fcsybase1_ctrl         +
* Monitor Mode                                       [Long-running monitoring> +
* Processes to Monitor                               [dataserver]
* Process Owner                                      [sybase]
  Instance Count                                     [1]                      #
* Stabilization Interval                             [120]                    #
* Restart Count                                      [3]                      #
  Restart Interval                                   [400]                    #
* Action on Application Failure                      [fallover]              +
  Notify Method                                      [/home/sybase/dataserver_down.sh>
#if notify only...this also notifies prior to failover
  Cleanup Method                                     [/home/sybase/ASE_HA.sh failover>
/
 Restart Method                                     [/home/sybase/ASE_HA.sh monitor>  /
```

Appendix C: Resources

The following websites provide useful references to supplement the information contained in this paper:

- IBM Power Systems AIX Information Center (all topics)
  - http://publib.boulder.ibm.com/infocenter/aix/v7r1/index.jsp (AIX 7.1)
  - http://publib.boulder.ibm.com/infocenter/aix/v6r1/index.jsp (AIX 6.1)
  - http://publib.boulder.ibm.com/infocenter/pseries/v5r3/index.jsp (AIX 5.3)
- Power Systems on IBM PartnerWorld®
  ibm.com/partnerworld/wps/pub/overview/news/B5P00PW
- AIX on IBM PartnerWorld
  ibm.com/partnerworld/wps/pub/overview/OE600
- IBM Publications Center
  www.elink.ibmlink.ibm.com/public/applications/publications/cgibin/pbi.cgi?CTY=US
- IBM Redbooks®
  ibm.com/redbooks
- Redbooks most relevant to the white paper *Optimizing SAP ASE on IBM AIX*:
  - AIX Version 5.3 differences guide
    ibm.com/redbooks/redbooks/pdfs/sg247463.pdf
  - AIX Version 6.1 differences guide
    ibm.com/redbooks**/**redbooks/pdfs/sg247559.pdf
  - AIX Version 7.1 differences guide
    ibm.com/redbooks/redbooks/pdfs/sg247910.pdf
  - IBM PowerVM Virtualization Introduction and Configuration
    ibm.com/redbooks/redbooks/pdfs/sg247940.pdf
  - IBM PowerVM Virtualization Managing and Monitoring
    ibm.com/redbooks/abstracts/sg247590.html
- IBM developerWorks®
  ibm.com/developerworks/aix/
- IBM Techdocs technical white papers (including this white paper)
  ibm.com/support/techdocs/atsmastr.nsf/Web/TechDocs
- SAP ASE general information
  http://www.sybase.com/products/informationmanagement/adaptiveserverenterprise
- SAP ASE 15.7 manuals
  http://infocenter.sybase.com/help/topic/com.sybase.infocenter.help.ase.15.7/title.htm

- SAP ASE 15.7 System Administration Guides
  - http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc31654.1570/html/sag1/title.htm
  - http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc31644.1570/html/sag2/title.htm
- SAP ASE 15.7 Performance and Tuning Manuals
  - http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc20020.1570/html/basics/title.htm
  - http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc00976.1570/html/statistics/title.htm
  - http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc00938.1570/html/locking/title.htm
  - http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc00842.1570/html/spsysmon/title.htm
  - http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc00848.1570/html/monitor_tables/title.htm
  - http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc00841.1570/html/phys_tune/title.htm
  - http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc00743.1570/html/queryprocessing/title.htm

Acknowledgements

Peter H. Barnett (IBM Corporation), Jian Yang (SAP AG), Deepak Narayana (IBM Corporation), and Shawn Bodily(IBM Corporation) authored this paper. It was built on research from the authors and their IBM and SAP colleagues.

The authors welcome readers to share their experiences and to provide feedback at: narayana@us.ibm.com (Deepak Narayana) phbarn@us.ibm.com (Peter H Barnett), jian.yang@sap.com (Jian Yang), sbodily@us.ibm.com (Shawn Bodily).

@IBMSystemsISVs

Trademarks and special notices