ExpressCluster® X *for Linux*

VMware vSphere™ 4 System Configuration Guide

02/26/2010
Third Edition

Revision History

Edition	Revised Date	Description
First	07/14/2009	New manual
Second	08/21/2009	Correspond to the internal version 2.1.1-1
Third	02/26/2010	- Described the procedure for adding a second virtual NIC - Described the procedure for associating a physical NIC with a virtual NIC - Described the firewall settings of vSphere - Described the settings for detecting an NIC error on a host - Described the settings of userw

Disclaimer

Information in this document is subject to change without notice.
NEC Corporation is not liable for technical or editorial errors or omissions in the information in this document.
You are completely liable for all risks associated with installing or using the product as described in this manual to obtain expected results and the effects of such usage.
The information in this document is copyrighted by NEC Corporation.
No part of this document may be reproduced or transmitted in any form by any means, electronic or mechanical, for any purpose, without the express written permission of NEC Corporation.

Trademark Information

ExpressCluster® X is a registered trademark of NEC Corporation.
AMD and AMD Virtualization are trademarks of Advanced Micro Devices, Inc.
Intel, Pentium, Xeon, and Intel VT are registered trademarks or trademarks of Intel Corporation.
VMware and VMotion are registered trademarks of VMware, Inc. in the United States.
Linux is a registered trademark of Linux Torvalds in the United States and other countries.
Other product names, company names, and products written in this manual are trademarks or registered trademarks of their respective companies.

Table of Contents

Preface

Who Should Use This Guide

This manual is intended for administrators who want to build a cluster system, system engineers who want to provide user support, and maintenance personnel.
This manual introduces software whose operation in an ExpressCluster environment has been checked. The software and setup examples introduced here are for reference only. They are not meant to guarantee the operation of each software product.

Scope of Application

This manual covers the ExpressCluster versions below.

- Host
  - ExpressCluster X 2.1 for Linux

- Guests
  - ExpressCluster X 2.1 for Windows
  - ExpressCluster X 2.0 for Windows
  - ExpressCluster X 2.1 for Linux
  - ExpressCluster X 2.0 for Linux

How This Guide Is Organized

Section I **Virtualization software**

Chapter 1 "VMware vSphere 4": Describes the procedures for and provides notes on using ExpressCluster in a VMware vSphere environment.

ExpressCluster X Documentation Set

The ExpressCluster manuals consist of the three guides below. The title and purpose of each guide is described below:

ExpressCluster X Getting Started Guide
This guide is intended for all users. The guide covers topics such as product overview, system requirements, and known problems.

ExpressCluster X Installation and Configuration Guide
This guide is intended for system engineers and administrators who want to build, operate, and maintain a cluster system. Instructions for designing, installing, and configuring a cluster system with ExpressCluster are covered in this guide.

ExpressCluster X Reference Guide
This guide is intended for system administrators. The guide covers topics such as how to operate ExpressCluster, function of each module, maintenance-related information, and troubleshooting. The guide is supplement to the *ExpressCluster X Installation and Configuration Guide*.

Conventions

In this guide, **Note**, **Important**, **Related Information** are used as follows:

Note:
Used when the information given is important, but not related to the data loss and damage to the system and machine.

Important:
Used when the information given is necessary to avoid the data loss and damage to the system and machine.

Related Information:
Used to describe the location of the information given at the reference destination.

The following conventions are used in this guide.

Convention	Usage	Example
Bold	Indicates graphical objects, such as fields, list boxes, menu selections, buttons, labels, icons, etc.	In **User Name**, type your name. On the **File** menu, click **Open Database**.
Angled bracket within the command line	Indicates that the value specified inside of the angled bracket can be omitted.	`clpstat –s[-h host_name]`
Monospace `(courier)`	Indicates path names, commands, system output (message, prompt, etc), directory, file names, functions and parameters.	`c:\Program files\EXPRESSCLUSTER`
Monospace bold `(courier)`	Indicates the value that a user actually enters from a command line.	Enter the following: **`clpcl -s -a`**
Monospace italic `(courier)`	Indicates that users should replace italicized part with values that they are actually working with.	`clpstat –s [-h host_name]`

Contacting NEC

For the latest product information, visit our website below:

http://www.nec.com/global/prod/expresscluster/

Section I Virtualization Software

This section explains the settings necessary for and provides notes on associating the virtualization software with ExpressCluster.

Chapter 1 VMware vSphere 4

Terms used in this chapter

The meanings of terms used in this chapter are provided below.

Term	Abbreviation	Explanation
Physical server	SV	Server on which VMware ESX or another OS is running
Standalone OS	Operating system	Normal OS used on its own, not with a virtualization platform
Host OS	Host	OS installed on a physical server as the virtualization platform, such as VMware ESX
Virtual machine	VM	Virtual server or client created on a host OS
Guest OS	Guest	OS installed on a virtual machine
ExpressCluster X	CLS	ExpressCluster X
ExpressCluster X SingleServerSafe	SSS	ExpressCluster X SingleServerSafe
Application	AP	Business application

Functional overview

By combining VMware vSphere and ExpressCluster, clusters that have the configurations below can be set up.

Inter-host OS clusters

ExpressCluster X is installed on the VMware ESX Service Console to cluster physical servers together. Both normal applications and guest OSs can be failed over.

By associating guests with hosts, applications in guest OSs can be monitored.

For details about how to set up the cluster, see “Checking the operation of an inter-host OS cluster” on page 23. If using a guest-to-host association, see “Using a guest-to-host association in an inter-host OS cluster” on page 26.

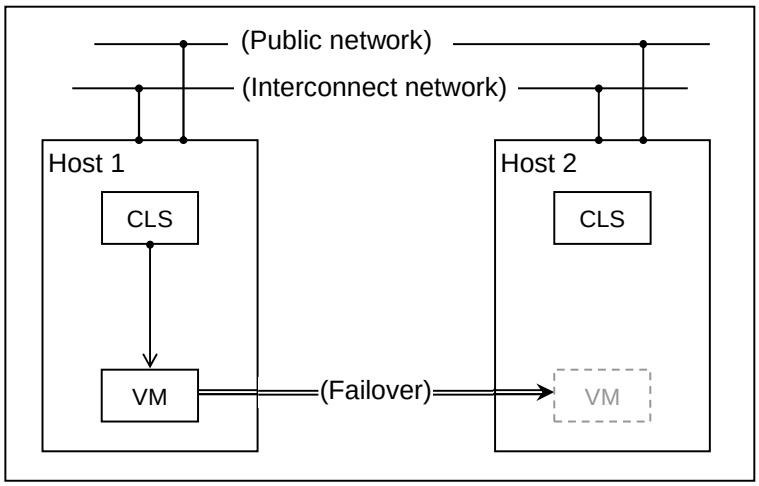

Figure 1: Outline of an inter-host OS cluster

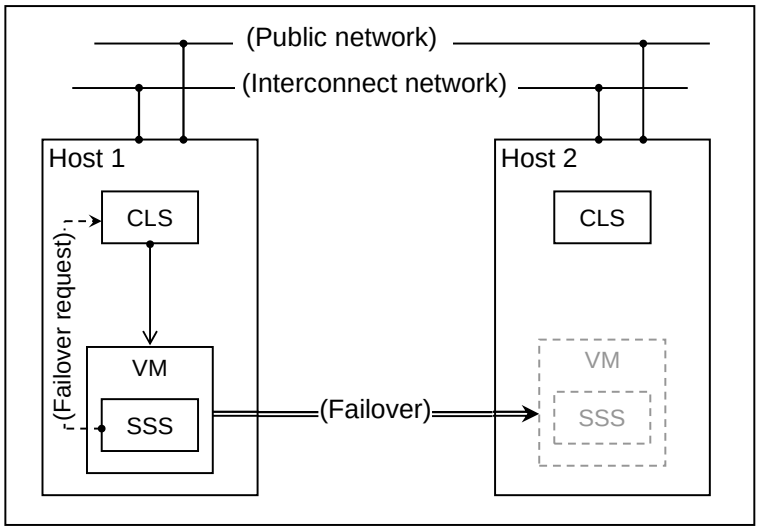

Figure 2: Outline of an inter-host OS cluster when guest-to-host associations are used

Inter-guest OS clusters

ExpressCluster X is installed on guest OSs to cluster virtual machines together. As in normal cluster systems, applications can be failed over, improving operational availability.

For details about how to set up the cluster, see “Setting up an inter-guest OS cluster” on page 27.

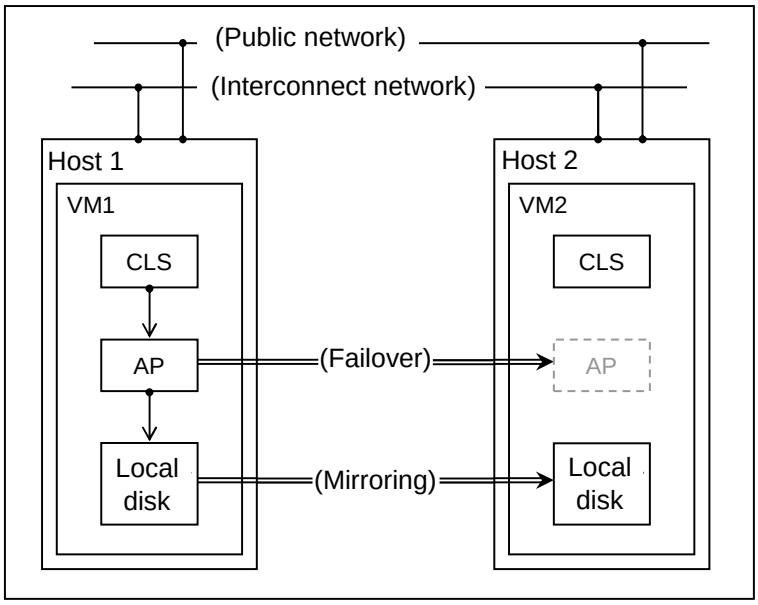

Figure 3: Outline of an inter-guest OS cluster of the mirror disk type

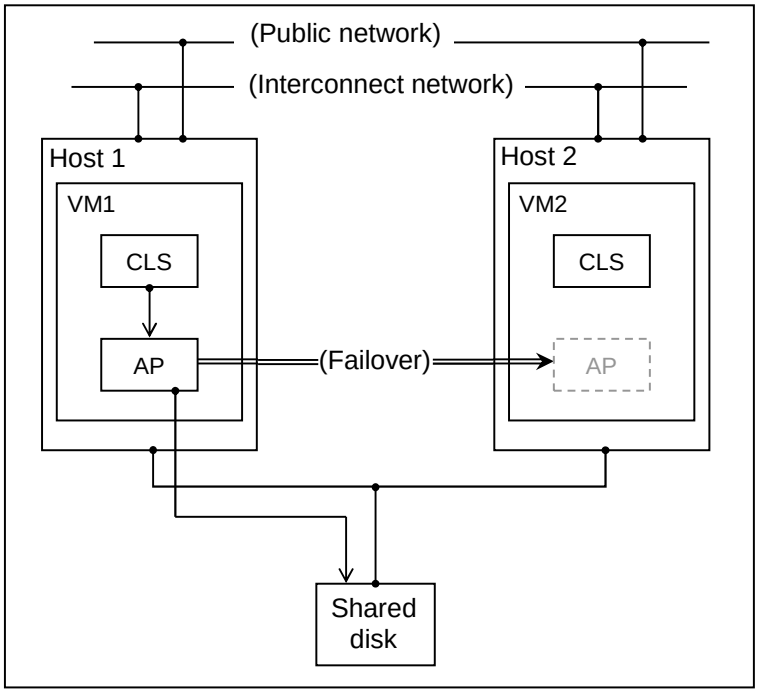

Figure 4: Outline of an inter-guest OS cluster of the shared disk type

Physical server-to-virtual machine cluster

ExpressCluster X is installed on the OS running on a physical server and on the OS running on a virtual machine to cluster the physical server and virtual machine together. As in normal cluster systems, applications can be failed over.

For details about how to set up the cluster, see “Setting up a physical server-to-virtual machine cluster” on page 29.

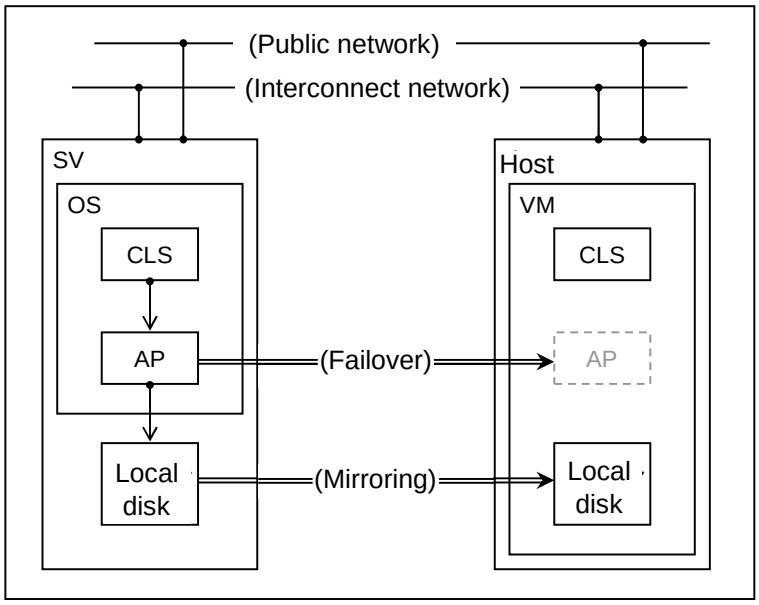

Figure 5: Outline of a physical server-to-virtual machine cluster of the mirror disk type

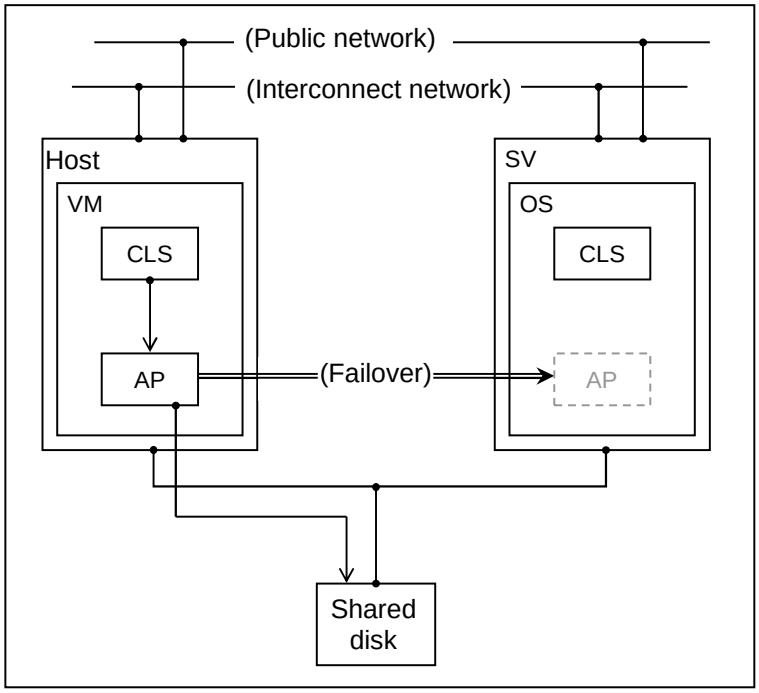

Figure 6: Outline of a physical server-to-virtual machine cluster of the shared disk type

VMware HA associations

A virtual machine sets up VMware HA, and, if a guest OS fails or the vCenter-to-ESX network is disconnected, the guest OS is failed over.

By installing ExpressCluster X SingleServerSafe on the VMware ESX Service Console, physical hardware errors that cannot be detected by VMware can be detected. If, in particular, an error is detected in the shared disk in which a guest OS is stored, ESX is shut down immediately so operations do not continue on the guest OS, which runs unstably. (Figure 7)

Also, by installing ExpressCluster X on a guest OS, guest OS errors (including virtual hardware errors and application errors) can be detected, so operations can be failed over. (Figure 8)

For details about how to set up this configuration, see “Associating with VMware HA” on page 31.

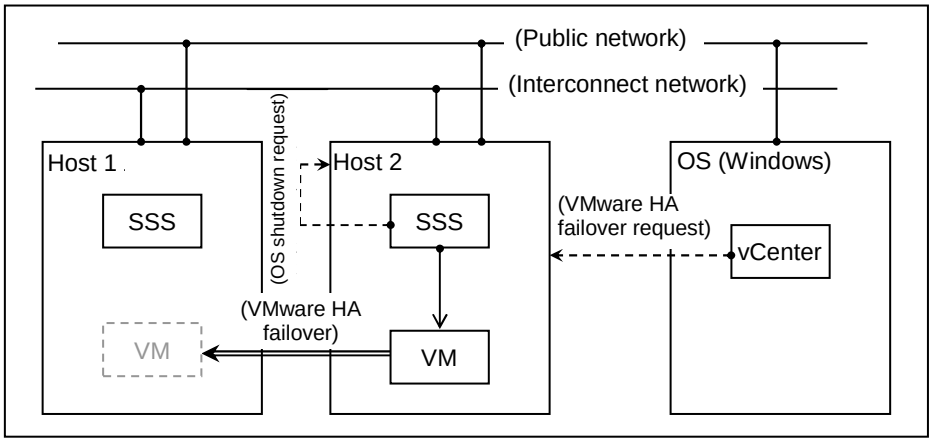

Figure 7: Outline of the configuration of associations between ExpressCluster X SingleServerSafe on hosts and VMware HA

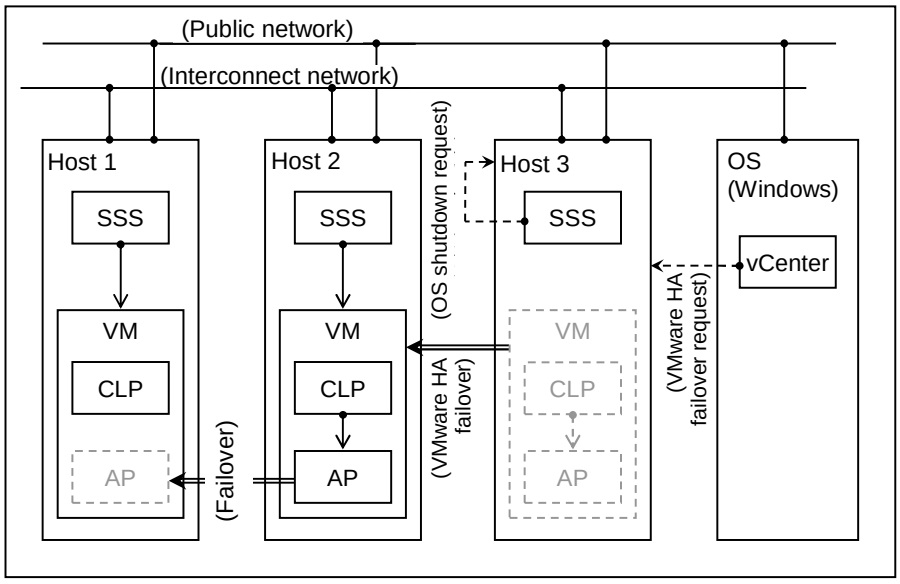

Figure 8: Outline of the configuration of associations among ExpressCluster X on guests, ExpressCluster X SingleServerSafe on hosts, and VMware HA

Operating environment

- The versions of VMware ESX and ExpressCluster covered in this chapter are as follows:

  VMware ESX

  - VMware ESX 4.0

  ExpressCluster (on hosts)

  - ExpressCluster X 2.1 for Linux
  - CLUSTERPRO X SingleServerSafe 2.1 for Linux

  ExpressCluster (on guests)

  - Supports guest OSs

Notes

Notes on inter-host OS clusters

- In VMware ESX 4.0 Service Console, install the x86_64 version of ExpressCluster (ExpressCluster-2.1.X-X.x86_64.rpm). Be careful not to install ExpressCluster for VMware ESX 3.5 (ExpressCluster-2.1.X-X.vmware.i386.rpm).
- An inter-host OS cluster is not made up of multiple virtual machines, and cannot be used for rolling maintenance on the OS or applications installed on the virtual machine (during which updates and patches are applied to the standby system while operations continue on the active system).
- In an inter-host OS cluster, the following cannot be used:
  - Mirror disk resource (md)
- If the internal version of ExpressCluster is 2.1.0-1 or earlier, the following cannot be used in an inter-host OS cluster:
  - Kernel mode LAN heartbeat resource (lankhb)
  - User space monitoring resource (userw)
  - Shutdown stalling monitoring
- If using the user space monitoring resource (userw) or shutdown stalling monitoring in an inter-host OS cluster, select **keepalive** as the monitoring method.
- If using a guest-to-host association, ExpressCluster X SingleServerSafe must be installed in the guest OS.

Notes on inter-guest OS clusters

- To use a shared disk in this configuration, you must set the **Share the SCSI bus** setting of the SCSI controller of the virtual machine to **Physical** or **Virtual**. Note that, if you select **Virtual**, you cannot set up an inter-guest OS cluster across an ESX host as shown in Figure 4.

  Physical: Virtual disks can be shared among the VMs on all ESX hosts.
  Virtual: Virtual disks can be shared among the VMs on a single ESX host.
- During cluster operation, do not suspend a virtual machine by using **Suspend**. If you suspend a virtual machine by using **Suspend**, ExpressCluster will detect a heartbeat timeout, and a failover group will be activated on another server. If, in this state, you use **Resume** to resume the virtual machine suspended using **Suspend**, both systems become active, and both virtual machines on which the failover group is running are shut down to protect data.

Notes on physical server-to-virtual machine clusters

- To use a shared disk in this configuration, you must set the shared disk type of the virtual machine to **Raw device mapping**.

Notes on using VMotion at the same time

- You must place the configuration file and disk image of the virtual machine on a shared disk.
- The VMware specifications prevent using VMotion when the **Share the SCSI bus** setting for the SCSI controller of the virtual machine is not **None**. This means that you cannot use VMotion and an inter-guest OS cluster that uses the shared disk at the same time.

Table 1: Whether VMotion can be used at the same time

	Shared disk type	Mirror disk type
Inter-host OS cluster	No	-
Inter-guest OS cluster	No	Yes
Physical server-to-virtual machine cluster	Yes	Yes

Notes on a VMware HA association configuration

- In this configuration, you cannot operate a virtual machine that stores a guest OS at a location other than a shared disk.
- In this configuration, VMotion and other applications are used, so a shared disk cannot be used on a virtual machine.
- If the internal version is 2.1.0-1 or earlier, the instance of ExpressCluster X SingleServerSafe on the VMware ESX Service Console cannot use the following:
  - Kernel mode LAN heartbeat resource (lankhb)
  - User space monitoring resource (userw)
  - Shutdown stalling monitoring
- When using the user space monitoring resource (userw) or shutdown stalling monitoring in the instance of ExpressCluster X SingleServerSafe on the VMware ESX Service Console, select keepalive as the monitoring method.
- To use VMware HA, the following conditions must be met:
  - A VMware vCenter Server is installed.
  - Multiple VMware ESX instances are installed, and each ESX instance has a VMware HA license.
  - A shared storage device is set up, and a virtual machine is stored in a data store in the shared storage. (Fibre Channel SAN, iSCSI SAN, or NFS can be used to connect the shared storage device.)
  - The DNS is set up correctly, and vCenter and all ESX service consoles allow name resolution with each other.
  - The host name specified for each ESX instance matches the corresponding host name registered in the DNS.
  - In each ESX instance, the Gateway address of the Service Console is specified correctly, and the Gateway responds to ping.
  - To enable virtual machine monitoring with VMware HA, VMware Tools are installed in the virtual machines to be monitored.

Notes on using ExpressCluster on the Service Console

- To use the second physical adapter, you must create a virtual switch to be linked with the target physical adapter when adding a network to the host network configuration.
- To use VMotion, you must add **VMkernel port** to the host network configuration.
- To use the second NIC in the Service Console, you must add **Service console port** to the host network configuration.
- You cannot specify IP addresses shared by ports in the host network configuration. To use **VMkernel** and **Service Console** at the same time, specify separate IP addresses.

Notes on using ExpressCluster on a virtual machine

- To use the second physical adapter, you must create a virtual switch to be linked with the target physical adapter when adding a network to the host network configuration.
- To use two NICs linked to different virtual switches on a virtual machine, you must add **Virtual machine's port group** to the host network configuration and set up the virtual switch corresponding to the **Network label** value of **Network adapter**.

Installing VMware ESX 4.0

Follow the installation guide provided by VMware, Inc. to install VMware ESX 4.0.

- VMware support resource document
  http://www.vmware.com/jp/support/pubs/

When using ExpressCluster on the Service Console, set up the firewall so that the communication ports of ExpressCluster can be accessed.

To disable the firewall:

```
# esxcfg-firewall --allow Incoming --allow Outgoing
```

For details about the port numbers used by ExpressCluster, refer to the following:

- *ExpressCluster X 2.1 for Linux Getting Started Guide*
  Chapter 5 Notes and Restrictions
  > Before installing ExpressCluster and after installing OS
  >> Communication port number

Adding the second virtual switch

For a server that contains two physical NICs, only a virtual switch related to one of them is created by default. Therefore, to use the other physical NIC, you must add another virtual switch.

Addition procedure:

(1) Make a connection from VMware vSphere Client, select a host (physical server), and then click **Network** on the **Configuration** tab.

(2) Click **Add network**.

(3) Select **Virtual machine**, and then click **Next**.

(4) Select **Create a virtual switch**, and then click **Next**.

(5) Enter any name in **Network label** of the port group properties, and then click **Next**.

(6) Check the settings, and then click **Finish**.

To use VMotion by way of the second virtual switch, execute **Add VMkernel** as described below.

(1) Click **Properties** of the second virtual switch, and then select the **Port** tab.

(2) Click **Add**, select **VMkernel**, and then click **Next**.

(3) Enter any name in **Network label** of the port group properties, select **Use this port group in VMotion**, and then click **Next**.

(4) Select **Use the next IP setting**, enter values in **IP address** and **Subnet mask**, and then click **Next**.

(5) Check the settings, and then click **Finish**.

(6) If the following warning is output, click **No**: “The default gateway is not set. To use this network interface, you may need to get the default gateway. Do you want to set it up it right now?”

(7) Click **Close** in **Properties** of the virtual switch.

To use the Service Console by way of the second virtual switch, execute **Add service console** as described below.

Note: Different IP settings must be specified for **For VMkernel** and **For Service Console**. Click **Properties** of the second virtual switch, and then select the **Port** tab.

(1) Click **Add**, select **Service Console**, and then click **Next**.

(2) Enter any name in **Network label** of the port group properties, and then click **Next**.

(3) Select **Use the next IP setting**, enter values in **IP address** and **Subnet mask**, and then click **Next**.

(4) Check the settings, and then click **Finish**.

(5) Click **Close** in **Properties** of the virtual switch.

Creating virtual machines

Create virtual machines according to the cluster system to be set up. Examples of creating a virtual machine are described below for reference.

For details about creating virtual machines, refer to the *Basic System Management Guide* provided by VMware, Inc.

- VMware support resource document
  http://www.vmware.com/jp/support/pubs/

Creating a virtual machine that uses a shared disk[1]

(1) Execute **Create a new virtual machine**.

(2) Select **Standard** from **Configuration**, and then click **Next**.

(3) Enter any name, and then click **Next**.

(4) Select an appropriate host, and then click **Next**.

(5) Select an appropriate data store, and then click **Next**.

(6) Select the guest OS to install, and then click **Next**.

(7) Enter any virtual disk size, and then click **Next**. In this step, specify the size of the disk in which the OS is to be installed. Specify the settings for the shared disk of the cluster system later.

(8) Check the settings, and then click **Finish**.

(9) Right-click the added virtual machine, and then execute **Edit settings**.

(10) Click **Add** on the **Hardware** tab.

(11) To add the second VM Network, select **Ethernet adapter**, and then click **Next**.

(12) Specify **Adapter type**, select **Network label of the second virtual switch**, and then click **Next**.

(13) Check the settings, and then click **Finish**.

(14) Select **Hard disk**, and then click **Next**.

(15) Select **Create a new virtual disk** or **Raw device mapping**, and then click **Next**. To set up a physical server-to-virtual machine cluster, be sure to select **Raw device mapping**.
For the second or subsequent virtual machine, select **Use the existing virtual disk**, click **Next**, and then select the ***.vmdk** file created for the first one.

(16) If you selected **Create a new virtual disk**, specify the size of the virtual disk. Select **Support clustering functions such as fault tolerance** of disk provisioning, and then select a data store on the shared disk.
If you selected **Raw device mapping**, specify the disk LUN to use. Next, click **Next**.

(17) For **Virtual device node**, select a node whose X value in **SCSI (X:Y)** or **IDE (X:Y)** differs from that of the disk for installing OSs, and then click **Next**.

(18) Check the settings, and then click **Finish**.

(19) Make sure that **SCSI controller** has been added to the hardware list, in addition to **Hard disk**. If the SCSI controller is not found, the **Virtual device node** setting of the added **Hard disk** might be incorrect.

[1] If you set up a shared disk on a virtual machine, you can no longer use VMotion. For details, see “Notes on using VMotion at the same time” on page 17.

(20) Select **SCSI controller**, change **Share the SCSI bus** to **Physical** or **Virtual**, and then click **OK**.

(21) A virtual machine that uses a shared disk is now created. If you have not installed an OS, install one.

Creating a virtual machine that uses a mirror disk

(1) Follow the steps up to (14) in "Creating a virtual machine that uses a shared disk".

(2) Select **Create a new virtual disk**, and then click **Next**. Also select **Create a new virtual disk** for the second and subsequent machines.

(3) If you selected **Create a new virtual disk**, specify the virtual disk size, and then click **Next**. It is recommended that the same size be specified for all virtual machines.

(4) Click **Next** without specifying anything.

(5) Check the settings, and then click **Finish**.

(6) A virtual machine that uses a mirror disk is now created. If you have not installed an OS, install one.

Setting up clusters

In the procedures below, the following failover group name, resource name, and monitor resource name are assumed. Change these names as appropriate according to your environment.

Group/resource/monitor resource	Name:
Failover group	failover-vm
Disk resource	sd-vm
exec resource	exec-vm
Custom monitor resources	genw-vm

Setting up an inter-host OS cluster

(1) Install ExpressCluster X on each host OS as described in the *ExpressCluster X Installation and Configuration Guide*[1]. The RPM of the x86_64 version is installed here.
If using the user space monitor resource and shutdown stalling monitoring, set the monitoring method to **keepalive**.

(2) Create a failover group that has a disk resource (sd-vm) as described in the *ExpressCluster X Installation and Configuration Guide*. Make sure that the failover group can be activated without a problem.

(3) On the server containing the failover target virtual machine, activate the failover group.

(4) If you have not created a virtual machine, create a virtual machine as described in the previous section, “Creating virtual machines”.

(5) Before activating the cluster, execute the command below on the VMware ESX Service Console containing the failover target virtual machine to unregister the virtual machine. Before unregistering the virtual machine, turn it off.
```
# vmware-cmd -s unregister vm_path
```

(6) Start the ExpressCluster Builder.

(7) Create a script for starting and stopping the virtual machine from ExpressCluster. Create vmpower.start.pl, vmpower.stop.pl, and clpvmmon.pl as described in the sample scripts at the end of this chapter, and save them anywhere on the machine on which the ExpressCluster Builder is running.

(8) Use the ExpressCluster Builder to add the exec resource to the failover group, and then edit this resource.

A) In the ExpressCluster Builder window, right-click the failover group name (failover-vm) in the tree view, and then click **Add Resource**.
B) From **Type**, select **execute resource**. Enter the name (exec-vm) for the exec resource, and then click **Next**.
C) Select **Start script**, and then click **Replace**. When a file selection window is displayed, select vmpower.start.pl, and replace start.sh.
D) Select **Stop script**, and then click **Replace**. When a file selection window is displayed, select vmpower.stop.pl, and replace stop.sh.

[1] Available from http://www.nec.com/global/prod/expresscluster/.

(9) Add a custom monitor resource by using the ExpressCluster Builder.

A) In the ExpressCluster Builder window, right-click **Monitors** in the tree view, and then click **Add Monitor Resource**.

B) From **Type**, select **custom monitor resource**. Enter any name (genw-vm) for the custom monitor resource, and then click **Next**.

C) Click **Replace**, and then replace genw.sh with clpvmmon.pl.

D) Change **Monitor Type** to **Asynchronous**, and then click **Next**.

E) As **Monitor Timing**, select **Active**, and then click **Browse**. When a list of resources that can be selected is displayed, select the previously created exec resource (exec-vm)[1]. Make sure that the exec resource is specified as the target resource, and then click **Next**.

F) As **Recovery Target**, select the name (failover-vm) of the failover group to which the exec resource (exec-vm) belongs, and then click **OK**. Edit parameters such as **Reactivation Threshold** as appropriate[2], and then click **Finish**.

(10) Stop the failover group (failover-vm) by using either the WebManager or clpgrp command.

(11) Suspend the cluster by using either the WebManager or clpcl command.

(12) Upload the configuration data created using the ExpressCluster Builder. From **File**, which is on the Builder menu, select **Upload the Configuration File** to upload the configuration data.

[1] When this setting is made, guest OS monitoring is started after start of the exec resource, i.e., after the start of the guest OS. Guest OS monitoring is stopped at the start of the stopping process on the exec resource, i.e., at the start of the saving of the guest OS.

[2] Adjust each parameter as appropriate as described in Chapter 6, "Monitor resource details" in the *ExpressCluster X Reference Guide*.

Checking the operation of an inter-host OS cluster

(1) Resume the cluster by using either the WebManager or clpcl command.

(2) Activate the failover group by using either the WebManager or clpgrp command. Make sure that the guest OS is running on the server on which the failover group is activated.

(3) Move the failover group by using either the WebManager or clpgrp command. Make sure that the guest OS is running on the server to which the failover group is moved.

(4) Shut down or reboot the physical server on which the failover group is running by using either the WebManager or clpdown command. At this time, make sure that the failover group has moved to another server and the guest OS is running.

(5) Shut down the guest OS, and then make sure that genw-vm detects an error and reactivates the recovery target or performs a failover. Make sure that the guest OS is restarted after the failover.

(6) Turn off the physical server from other than ExpressCluster, and make sure that the other server detects the stoppage of the server, activates the failover group, and that the guest OS is restarted.

(7) In addition to the above, implement the items described in “Operation tests”, in Chapter 8, “Verifying operation” in the *ExpressCluster X Installation and Configuration Guide*, as appropriate.

Using a guest-to-host association in an inter-host OS cluster

(1) Create the cluster configuration data by using the Builder (as described in "Setting up an inter-host OS cluster" on page 23).

(2) Install ExpressCluster X SingleServerSafe in the guest OS as described in the *ExpressCluster X SingleServerSafe Installation and Configuration Guide*[1].

(3) Edit the cluster configuration data in the guest OS by using the ExpressCluster Builder. For details about scripts, see the Appendix.

A) Add the monitor resources to be monitored (such as the pid, appli, and oracle monitor resources).

B) Enable **Execute Script before Final Action** for the monitor resources, and then select **Settings**.

C) If the guest OS is Linux, click **Replace** to replace the contents of preaction.sh with those of vmpreaction.sh.
If the guest OS is Windows, click **Replace** to replace the contents of preaction.bat with those of vmpreaction.bat.

D) Specify other settings as appropriate.

(4) Suspend the cluster by using either the WebManager or clpcl command.

(5) Upload the configuration data created using the ExpressCluster Builder. From **File**, which is on the Builder menu, select **Upload the Configuration File** to upload the configuration data.

[1] Available from http://www.nec.co.jp/clusterpro/sss/sss_index.html.

Setting up an inter-guest OS cluster

(1) If you have not created a virtual machine, create one as described in “Creating virtual machines”.

(2) Install a guest OS supported by ExpressCluster on the virtual machine.

(3) Install ExpressCluster in the guest OS as described in the *ExpressCluster X Installation and Configuration Guide*.

(4) Set up a cluster by using the ExpressCluster Builder and following the *ExpressCluster X Installation and Configuration Guide*.

(5) Upload the configuration data created using the ExpressCluster Builder. From **File**, which is on the Builder menu, select **Upload the Configuration File** to upload the configuration data.

Checking the operation of an inter-guest OS cluster

(1) Activate the cluster by using either the WebManager or clpcl command.

(2) Move the failover group by using either the WebManager or clpgrp command. Make sure that the failover group is running on the server to which the failover group is moved by using either the WebManager or clpstat command.

(3) Shut down or reboot the virtual machine on which the failover group is running by using either the WebManager or clpdown command. At this time, make sure that the failover group is running on another server by using either the WebManager or clpstat command.

(4) Turn off the physical server from other than ExpressCluster, and make sure that the other server detects the stoppage of the server and activates the failover group by using either the WebManager or clpstat command.

(5) In addition to the above, implement the items described in "Operation tests", in Chapter 8, "Verifying operation" in the *ExpressCluster X Installation and Configuration Guide*, as appropriate.

Setting up a physical server-to-virtual machine cluster

(1) If you have not created a virtual machine, create one as described in “Creating virtual machines”.

(2) Install a guest OS supported by ExpressCluster on the virtual machine.

(3) Install ExpressCluster on the physical server and guest OS as described in the *ExpressCluster X Installation and Configuration Guide*.

(4) Set up a cluster by using the ExpressCluster Builder and following the *ExpressCluster X Installation and Configuration Guide*.

(5) Upload the configuration data created using the ExpressCluster Builder. From **File**, which is on the Builder menu, select **Upload the Configuration File** to upload the configuration data.

Checking the operation of a physical server-to-virtual machine cluster

(1) Activate the cluster by using either the WebManager or clpcl command.

(2) Move the failover group by using either the WebManager or clpgrp command. Make sure that the failover group is running on the server to which the failover group is moved by using either the WebManager or clpstat command.

(3) Shut down or reboot the server on which the failover group is running by using either the WebManager or clpdown command. At this time, make sure that the failover group is running on another server by using either the WebManager or clpstat command.

(4) Turn off the physical server from other than ExpressCluster, and make sure that the other server detects the stoppage of the server and activates the failover group by using either the WebManager or clpstat command.

(5) In addition to the above, implement the items described in "Operation tests", in Chapter 8, "Verifying operation" in the *ExpressCluster X Installation and Configuration Guide*, as appropriate.

Associating with VMware HA

Associating an instance of ExpressCluster X SingleServerSafe on a host with VMware HA

(1) Create as many virtual machines as necessary.

(2) Set up VMware HA from vCenter.

A) Right-click the data center in the inventory, and then select **New cluster** to create a cluster.

B) Drag and drop a host in the inventory to the created cluster to make the host participate in the cluster. Make all the hosts that are to be failover targets for the guest OS participate in the cluster.

C) Right-click the created cluster, and then select **Edit Settings** to display the cluster configuration window.

D) Select the left pane **Cluster Function** in the cluster configuration window, and then select **Enable VMware HA**.

E) If you require virtual machine monitoring by VMware HA, select the left pane **VMware HA** followed by **Monitor Virtual Machine** in the cluster configuration window, and then select **Enable Virtual Machine Monitoring** in the **Virtual Machine Monitoring Status** field.[1]

(3) Install ExpressCluster X SingleServerSafe (SSS) on the Service Console of VMware ESX as described in the *ExpressCluster X Installation and Configuration Guide*.

(4) Specify diskw for the installed SSS so that the shared disk in which the guest OS is stored is monitored. Select **READ(O_DIRECT)** as the diskw monitor type, and select **OS Shutdown** as the final action. If using the user space monitor resource and shutdown stalling monitoring, set the monitoring method to **keepalive**.

A) Add the monitor target monitor resources (such as the NIC Link up/down monitor resources).

B) Enable **Execute Script before Final Action** for the monitor resources, and then select **Settings**.

C) If the guest OS is Linux, click **Replace** to replace the contents of preaction.sh with those of vmpreaction.sh.
If the guest OS is Windows, click **Replace** to replace the contents of preaction.bat with those of vmpreaction.bat.

(5) Set up other monitors as appropriate.

[1] For this setting to take effect, you must install VMware Tools on the guest and operate them.

Associating an instance of ExpressCluster X on a guest and an instance of ExpressCluster X SingleServerSafe on a host with VMware HA

(1) Perform the steps up to (2) in the previous section, “Associating an instance of ExpressCluster X SingleServerSafe on a host with VMware HA”.

(2) Install ExpressCluster in the guest OS as described in the *ExpressCluster X Installation and Configuration Guide*.

(3) Set up a cluster by using the ExpressCluster Builder and following the *ExpressCluster X Installation and Configuration Guide*. If using the user space monitor resource and shutdown stalling monitoring, set the monitoring method to **keepalive**.

(4) Perform step (3) and the subsequent steps in the previous section, “Associating an instance of ExpressCluster X SingleServerSafe on a host with VMware HA”.

Appendix A Sample Scripts

This appendix contains samples of scripts necessary for setting up inter-host OS clusters. Edit the **bold** portions in the script as required according to your environment.

vmpower.start.pl

This is a script for starting a virtual machine.

```
#!/usr/bin/perl -w

#
# Script for power on the Virtual Machine
#

use strict;

#-------------------------------------------------------------------------------
# Configuration
#-------------------------------------------------------------------------------
# The path to VM configuration file. This must be absolute UUID-based path.
my $cfg_path = "/vmfs/volumes/4a276815-6acb55cc-e2a8-0019dbc0afcc/vm1/vm1.vmx";

# The interval to check the vm status. (second)
my $interval = 1;
# The miximum count to check the vm status.
my $max_cnt = 100;
# The timeout to start the vm. (second)
my $start_to = 10;
#-------------------------------------------------------------------------------
my $vmname = $cfg_path; # VMname to be outputted on log.
$vmname =~ s/^(.*\/)(.*)(\.vmx)/$2/;

# VM operation command path
my $vmcmd = "/usr/bin/vmware-cmd";

# VM execution state map
my %state = (
   "VM_EXECUTION_STATE_ON" => "on",
   "VM_EXECUTION_STATE_OFF" => "off",
   "VM_EXECUTION_STATE_SUSPENDED" => "suspended",
   "VM_EXECUTION_STATE_STUCK" => "stuck",
   "VM_EXECUTION_STATE_UNKNOWN" => "unknown"
);

#-------------------------------------------------------------------------------
# Main
#-------------------------------------------------------------------------------
exit 1 if (!&RegisterVm());

if (&IsPoweredOn()){
   exit 0;
}
else{
   if (&PowerOn()){
      if (&WaitPoweredOnDone()){
         exit 0;
      }
      else{
         exit 1;
      }
   }
   else{
```

```
      exit 1;
   }
}

#-------------------------------------------------------------------------------
# Functions
#-------------------------------------------------------------------------------
sub RegisterVm{
   my $svop = "-s register";
   my $vmcmd_list = $vmcmd . " -l";
   my @vmlist = `$vmcmd_list`;
   my $ret = 0;
   my $opn_ret;
   my $line;

   foreach (@vmlist){
      if (/$cfg_path/){
         &Log("[I] [$vmname] at localhost already registered.\n");
         return 1;
      }
   }

   $opn_ret = open(my $fh, $vmcmd . " " . $svop . " " . $cfg_path . " 2>&1 |");
   if (!$opn_ret){
      &Log("[E] [$vmname] at localhost: $vmcmd $svop could not be executed.\n");
      return 0;
   }

   $line = <$fh>;
   if (defined($line)){
      &Log("[E] [$vmname] at localhost: Could not register VM: $line\n");
   }
   else{
      $ret = 1;
      &Log("[I] [$vmname] at localhost: Registered.\n");
   }

   close($fh);

   return $ret;
}
#-------------------------------------------------------------------------------
sub IsPoweredOn{

   if (&IsEqualState($state{"VM_EXECUTION_STATE_ON"})){
      return 1;
   }
   else{
      return 0;
   }
}
#-------------------------------------------------------------------------------
sub IsEqualState{
   my $vmop = "getstate";
   my $state = shift;
   my $ret = 0;
   my $opn_ret;
   my $line;

   $opn_ret = open(my $fh, $vmcmd . " " . $cfg_path . " " . $vmop . " 2>&1 |");
   if (!$opn_ret){
      &Log("[E] [$vmname] at localhost: $vmcmd $vmop could not be executed.\n");
      return 0;
   }

   $line = <$fh>;
```

```
   if (defined($line)){
      chomp($line);
      if ($line =~ /^$vmop\(\)\s=\s(.+)$/){
         $ret = 1 if ($1 eq $state);
         &Log("[D] [$vmname] at localhost: VM execution state is $1.\n");
      }
      else{
         &Log("[E] [$vmname] at localhost: Could not get VM execution state: $line\n");
      }
   }

   close($fh);

   return $ret;
}
#--------------------------------------------------------------------------------
sub PowerOn{
   my $vmop = "start";
   my $ret = 0;
   my $opn_ret;
   my $line;

   $opn_ret = open(my $fh, $vmcmd . " " . $cfg_path . " " . $vmop . " 2>&1 |");
   if (!$opn_ret){
      &Log("[E] [$vmname] at localhost: $vmcmd $vmop could not be executed.\n");
      return 0;
   }

   eval{
      local $SIG{ALRM} = sub { die "timeout" };
      alarm($start_to);
      $line = <$fh>;
      alarm(0);
   };
   alarm(0);

   if ($@){
      if($@ =~ /timeout/){
         &Log("[E] [$vmname] at localhost: Cound not start VM: timeout($start_to
second)\n");

         if (&IsEqualState($state{"VM_EXECUTION_STATE_STUCK"})){
            $ret = 1 if (&ResolveVmStuck());
         }
      }
   }
   else{
      if (defined($line)){
         chomp($line);
         if ($line =~ /^$vmop\(\)\s=\s(.+)$/){
            if ($1 == 1){
               $ret = 1;
               &Log("[I] [$vmname] at localhost: Started.\n");
            }
            else{
               &Log("[E] [$vmname] at localhost: Cound not start VM: $1\n");
            }
         }
         else{
            &Log("[E] [$vmname] at localhost: Cound not start VM: $line\n");

            if (&IsEqualState($state{"VM_EXECUTION_STATE_STUCK"})){
               $ret = 1 if (&ResolveVmStuck());
            }
         }
      }
```

```
      close($fh);
   }

   return $ret;
}
#-------------------------------------------------------------------------------
sub ResolveVmStuck{
   my $vmop = "answer";
   my $ret = 0;
   my $opn_ret;
   my $line;

   $opn_ret = open(my $fh, "| ". $vmcmd . " " . $cfg_path . " " . $vmop);
   if (!$opn_ret){
      &Log("[E] [$vmname] at localhost: $vmcmd $vmop could not be executed.\n");
      return 0;
   }

   # Answering "1) I _moved it" to keep vm config.
   print($fh "1\n");

   close($fh);

   if (&IsEqualState($state{"VM_EXECUTION_STATE_STUCK"})){
      &Log("[E] [$vmname] at localhost: VM stuck could not be resolved.\n");
   }
   else{
      $ret = 1;
      &Log("[I] [$vmname] at localhost: VM stuck is resolved.\n");
   }

   return $ret;
}
#-------------------------------------------------------------------------------
sub WaitPoweredOnDone{

   for (my $i = 0; $i < $max_cnt; $i++){
      if (&IsEqualState($state{"VM_EXECUTION_STATE_ON"})){
         &Log("[I] [$vmname] at localhost: Powered on done. ($i)\n");
         return 1;
      }
      sleep $interval;
   }

   &Log("[E] [$vmname] at localhost: Not powered on done. ($max_cnt)\n");

   return 0;
}
#-------------------------------------------------------------------------------
sub Log{
   my ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) = localtime(time);
   $year += 1900;
   $mon += 1;
   my $date = sprintf "%d/%02d/%02d %02d:%02d:%02d", $year, $mon, $mday, $hour, $min,
$sec;
   print "$date $_[0]";
   return 0;
}
```

vmpower.stop.pl

This is a script for stopping a virtual machine. Edit the parts in **bold** as appropriate before use.

```
#!/usr/bin/perl -w

#
# Script for power off the Virtual Machine
#

use strict;

#--------------------------------------------------------------------------------
# Configuration
#--------------------------------------------------------------------------------
# The path to VM configuration file. This must be absolute UUID-based path.
my $cfg_path = "/vmfs/volumes/4a276815-6acb55cc-e2a8-0019dbc0afcc/vm1/vm1.vmx";

# The interval to check the vm status. (second)
my $interval = 5;
# The miximum count to check the vm status.
my $max_cnt = 100;
#--------------------------------------------------------------------------------
my $vmname = $cfg_path; # VMname to be outputted on log.
$vmname =~ s/^(.*\/)(.*)(\.vmx)/$2/;

# VM operation command path
my $vmcmd = "/usr/bin/vmware-cmd";

# VM execution state map
my %state = (
   "VM_EXECUTION_STATE_ON" => "on",
   "VM_EXECUTION_STATE_OFF" => "off",
   "VM_EXECUTION_STATE_SUSPENDED" => "suspended",
   "VM_EXECUTION_STATE_STUCK" => "stuck",
   "VM_EXECUTION_STATE_UNKNOWN" => "unknown"
);

#--------------------------------------------------------------------------------
# Main
#--------------------------------------------------------------------------------
if (&IsPoweredOn()){
   if (&PowerOff()){
      if (!&WaitPoweredOffDone()){
         exit 1;
      }
   }
   else{
      exit 1;
   }
}

if (&UnRegisterVm()){
   exit 0;
}
else{
   exit 1;
}

#--------------------------------------------------------------------------------
# Functions
#--------------------------------------------------------------------------------
sub IsPoweredOn{

   if (&IsEqualState($state{"VM_EXECUTION_STATE_ON"})){
```

```
      return 1;
   }
   else{
      return 0;
   }
}
#-------------------------------------------------------------------------------
sub IsEqualState{
   my $vmop = "getstate";
   my $state = shift;
   my $ret = 0;
   my $opn_ret;
   my $line;

   $opn_ret = open(my $fh, $vmcmd . " " . $cfg_path . " " . $vmop . " 2>&1 |");
   if (!$opn_ret){
      &Log("[E] [$vmname] at localhost: $vmcmd $vmop could not be executed.\n");
      return 0;
   }

   $line = <$fh>;
   if (defined($line)){
      chomp($line);
      if ($line =~ /^$vmop\(\)\s=\s(.+)$/){
         $ret = 1 if ($1 eq $state);
         &Log("[D] [$vmname] at localhost: VM execution state is $1.\n");
      }
      else{
         &Log("[E] [$vmname] at localhost: Could not get VM execution state: $line\n");
      }
   }

   close($fh);

   return $ret;
}
#-------------------------------------------------------------------------------
sub PowerOff{
   my $ret;

   # Soft stop.
   $ret = &PowerOffOpMode("soft");

   # Hard stop if Soft stop failed.
   if (!$ret){
      $ret = &PowerOffOpMode("hard");
   }

   return $ret;
}
#-------------------------------------------------------------------------------
sub PowerOffOpMode{
   my $vmop = "stop";
   my $powerop_mode = shift;
   my $ret = 0;
   my $opn_ret;
   my $line;

   return 0 if ($powerop_mode !~ /^hard|soft$/);

   $opn_ret = open(my $fh, $vmcmd . " " . $cfg_path . " " . $vmop . " " . $powerop_mode .
" 2>&1 |");
   if (!$opn_ret){
      &Log("[E] [$vmname] at localhost: $vmcmd $vmop $powerop_mode could not be
executed.\n");
      return 0;
```

```
   }

   $line = <$fh>;
   if (defined($line)){
      chomp($line);
      if ($line =~ /^$vmop\($powerop_mode\)\s=\s(.+)$/){
         if ($1 == 1){
            $ret = 1;
            &Log("[I] [$vmname] at localhost: Stopped. ($powerop_mode)\n");
         }
         else{
            &Log("[E] [$vmname] at localhost: Cound not stop ($powerop_mode) VM: $1\n");
         }
      }
      else{
         &Log("[E] [$vmname] at localhost: Cound not stop ($powerop_mode) VM: $line\n");
      }
   }
   else{
      if ($powerop_mode eq "soft"){
         $ret = 1;
         &Log("[I] [$vmname] at localhost: Stopped. ($powerop_mode)\n");
      }
   }

   close($fh);

   return $ret;
}
#--------------------------------------------------------------------------------
sub WaitPoweredOffDone{

   for (my $i = 0; $i < $max_cnt; $i++){
      if (&IsEqualState($state{"VM_EXECUTION_STATE_OFF"})){
         &Log("[I] [$vmname] at localhost: Powered off done. ($i)\n");
         return 1;
      }
      sleep $interval;
   }

   &Log("[E] [$vmname] at localhost: Not powered off done. ($max_cnt)\n");

   return 0;
}
#--------------------------------------------------------------------------------
sub UnRegisterVm{
   my $svop = "-s unregister";
   my $vmcmd_list = $vmcmd . " -l";
   my @vmlist = `$vmcmd_list`;
   my $ret = 0;
   my $opn_ret;
   my $flag = 0;
   my $line;

   foreach (@vmlist){
      if (/$cfg_path/){
         $flag = 1;
      }
   }

   if ($flag == 0){
      &Log("[I] [$vmname] at localhost already unregistered.\n");
      return 1;
   }
   else{
      $opn_ret = open(my $fh, $vmcmd . " " . $svop . " " . $cfg_path . " 2>&1 |");
```

```
      if (!$opn_ret){
         &Log("[E] [$vmname] at localhost: $vmcmd $svop could not be executed.\n");
         return 0;
      }

      $line = <$fh>;
      if (defined($line)){
         &Log("[E] [$vmname] at localhost: Could not unregister VM: $line\n");
      }
      else{
         $ret = 1;
         &Log("[I] [$vmname] at localhost: Unregistered.\n");
      }

      close($fh);
   }

   return $ret;
}
#------------------------------------------------------------------------------
sub Log{
   my ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) = localtime(time);
   $year += 1900;
   $mon += 1;
   my $date = sprintf "%d/%02d/%02d %02d:%02d:%02d", $year, $mon, $mday, $hour, $min,
$sec;
   print "$date $_[0]";
   return 0;
}
```

clpvmmon.pl

This is a script for checking the start state of a virtual machine. Edit the parts in **bold** as appropriate before use.

```
#!/usr/bin/perl -w

#
# Script for monitoring the Virtual Machine
#

use strict;

#------------------------------------------------------------------------------
# Configuration
#------------------------------------------------------------------------------
# The path to VM configuration file. This must be absolute UUID-based path.
my $cfg_path = "/vmfs/volumes/4a276815-6acb55cc-e2a8-0019dbc0afcc/vm1/vm1.vmx";

# The interval to check the vm status. (second)
my $interval = 1;
#------------------------------------------------------------------------------
my $vmname = $cfg_path; # VMname to be outputted on log.
$vmname =~ s/^(.*\/)(.*)(\.vmx)/$2/;

# VM operation command path
my $vmcmd = "/usr/bin/vmware-cmd";

# VM execution state map
my %state = (
   "VM_EXECUTION_STATE_ON" => "on",
   "VM_EXECUTION_STATE_OFF" => "off",
   "VM_EXECUTION_STATE_SUSPENDED" => "suspended",
   "VM_EXECUTION_STATE_STUCK" => "stuck",
   "VM_EXECUTION_STATE_UNKNOWN" => "unknown"
);
```

```
#-------------------------------------------------------------------------------
# Main
#-------------------------------------------------------------------------------
system("\"ulimit\" -s unlimited");
while (&IsHbIncrease()){
}
exit 0;

#-------------------------------------------------------------------------------
# Functions
#-------------------------------------------------------------------------------
sub IsHbIncrease{
   my $last_hb = -1;

   for (my $i = 0; $i < 2; $i++){
      if (&IsPoweredOn()){
         my $hb = &GetHb();
         if ($hb == -1){
            return 0;
         }

         if ($hb == $last_hb){
            &Log("[I] [$vmname] is stalled.\n");
            return 0;
         }
         else{
            $last_hb = $hb;
         }
      }
      else{
         &Log("[I] [$vmname] is powered off.\n");
         return 0;
      }
      sleep $interval;
   }

   return 1
}
#-------------------------------------------------------------------------------
sub GetHb{
   my $vmop = "getheartbeat";
   my $ret = -1;
   my $opn_ret;
   my $line;

   $opn_ret = open(my $fh, $vmcmd . " " . $cfg_path . " " . $vmop . " 2>&1 |");
   if (!$opn_ret){
      &Log("[E] [$vmname] at localhost: $vmcmd $vmop could not be executed.\n");
      return -1;
   }

   $line = <$fh>;
   if (defined($line)){
      chomp($line);
      if ($line =~ /^$vmop\(\)\s=\s(.+)$/){
         $ret = $1;
#        &Log("[D] [$vmname] at localhost: Got VM heartbeat count $1.\n");
      }
      else{
         &Log("[E] [$vmname] at localhost: Could not get VM heartbeat count: $line\n");
      }
   }

   close($fh);
```

```
   return $ret;
}
#-------------------------------------------------------------------------------
sub IsPoweredOn{

   if (&IsEqualState($state{"VM_EXECUTION_STATE_ON"})){
      return 1;
   }
   else{
      return 0;
   }
}
#-------------------------------------------------------------------------------
sub IsEqualState{
   my $vmop = "getstate";
   my $state = shift;
   my $ret = 0;
   my $opn_ret;
   my $line;

   $opn_ret = open(my $fh, $vmcmd . " " . $cfg_path . " " . $vmop . " 2>&1 |");
   if (!$opn_ret){
      &Log("[E] [$vmname] at localhost: $vmcmd $vmop could not be executed.\n");
      return 0;
   }

   $line = <$fh>;
   if (defined($line)){
      chomp($line);
      if ($line =~ /^$vmop\(\)\s=\s(.+)$/){
         $ret = 1 if ($1 eq $state);
#        &Log("[D] [$vmname] at localhost: VM execution state is $1.\n");
      }
      else{
         &Log("[E] [$vmname] at localhost: Could not get VM execution state: $line\n");
      }
   }

   close($fh);

   return $ret;
}
#-------------------------------------------------------------------------------
sub Log{
   my ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) = localtime(time);
   $year += 1900;
   $mon += 1;
   my $date = sprintf "%d/%02d/%02d %02d:%02d:%02d", $year, $mon, $mday, $hour, $min,
$sec;
   print "$date $_[0]";
   return 0;
}
```

vmpreaction.sh

This is a script for Linux that issues a failover request from an SSS instance on a guest to CLP on a host. Edit the parts in **bold** as appropriate before use.

```
#! /bin/sh
#***********************************************
#* preaction.sh *
#***********************************************
ulimit -s unlimited

# Write the name of the VM resource to be started by ESX.
CLPRSC="vmpower"
# Write the IP addresses of each ESX instance, delimited by commas.
CLPIP="10.0.0.1,10.0.0.2"

/opt/nec/clusterpro/bin/clptrnreq -t GRP_FAILOVER -r $CLPRSC -h $CLPIP
exit 0
```

vmpreaction.bat

This is a batch file for Windows that issues a failover request from an SSS instance on a guest to CLP on a host. Edit the parts in **bold** as appropriate before use.

```
rem ************************************
rem *          preaction.bat           *
rem ************************************

echo START
echo %CLP_MONITORNAME%

SET CLPRSC=vmpower
SET CLPIP=10.0.0.1,10.0.0.2

clptrnreq.exe -t GRP_FAILOVER -r %CLPRSC% -h %CLPIP%
echo EXIT
```