

Dennis Wunder, IBM  
Carlos Ordonez, IBM  
Michael MacIsaac, IBM

Chapter 1.

# z/VM VSWITCH with failover

An overview of z/VM's virtual switch is described in the first section, then a scenario, or "HOWTO", is presented in section 1.2, "Setting up VSWITCH with failover" on page 4.

## 1.1 Overview

z/VM 4.2 introduced a *Guest LAN* which allows virtual local area networks to exist within z/VM's Control Program. When utilizing a Guest LAN however, a z/VM or Linux machine has to act as a router to move TCP/IP packets to an outside LAN. This routing function can increase path length and CPU utilization.

While the Guest LAN is still a useful feature, z/VM 4.4 improved networking with the addition of a *virtual switch*. A z/VM virtual switch (referred to as VSWITCH hereafter) can be created as a special type of z/VM *Guest LAN*. It provides a network of virtual adapters to connect to an OSA-Express QDIO adapter, allowing Linux guests to connect through it to join the same physical network as the OSA-Express QDIO adapter.

### 1.1.1 VSWITCH controller

The management of the VSWITCH is provided by a *controller*. A controller is a z/VM TCP/IP virtual machine that handles the initialization and configuration tasks required for the OSA-Express VSWITCH connection. In order for a VSWITCH to provide connectivity to a LAN, at least one TCP/IP service machine must be configured to be a controller. For better availability, a backup VSWITCH controller can also be defined.

### 1.1.2 VSWITCH access

Access to a VSWITCH is controlled by granting or revoking access between it and each Linux guest that will use it. This can be done by the **CP SET VSWITCH** command, or via the **MODIFY VSWITCH** statement in the SYTEM CONFIG file.

In z/VM 5.1, security products such as VM:Secure from CA or RACF/VM from IBM will be able (with the necessary enhancements) to control VSWITCH access.

### 1.1.3 VSWITCH block diagrams

The VSWITCH is a virtual networking device that provides IP switching between an OSA-Express adapter and nodes connected to it. The connectivity provided through the VSWITCH makes the Guest LAN part of the physical LAN segment where the OSA adapter is connected. This means the IP addresses given to the Linux guests belong to the same network as the OSA card. The VSWITCH “owns” the OSA-Express control and data connections (read, write, and data devices) and provides a central point of control and management for IP traffic between the guest nodes and the OSA-Express adapter.

A block diagram of a VSWITCH is shown in Figure 1-1.

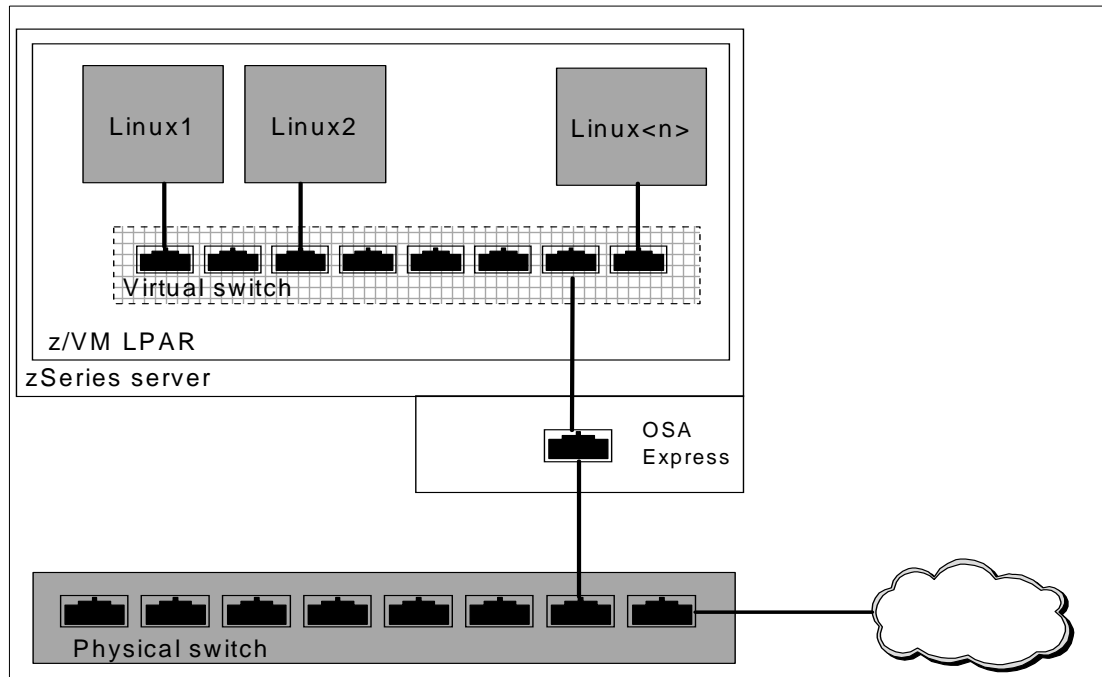


Figure 1-1 Virtual and physical switch block diagram

You can think also think of a VSWITCH as an extension of the external switch where the OSA adapter is connected. Even though all the IP communications will go through the OSA Express card, a conceptual depiction of the physical and virtual switches is shown in Figure 1-2.

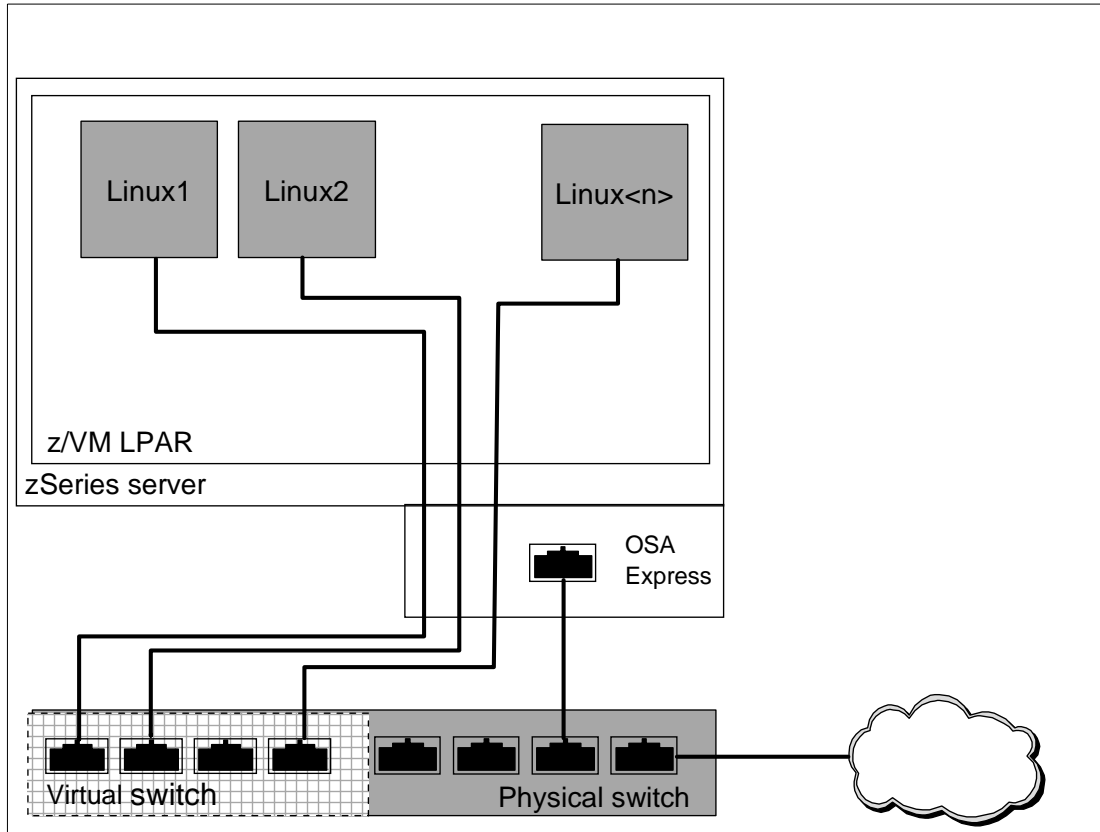


Figure 1-2 Virtual and physical switch conceptual block diagram

Each VSWITCH operates independently, but logically a VSWITCH or a portion of a VSWITCH can be defined to be part of a IEEE Virtual Local Area Network (also known as VLAN or IEEE 802.1Q). IEEE VLANs facilitate easy administration of logical groups of systems that can communicate as if they were on the same Local Area Network (LAN). VLANs are not described further in this paper, but more information is available in Chapter 7 in the z/VM manual: *Virtual Machine Operation*.

Deployment of a VSWITCH can reduce the CPU utilization cost and latency associated with a z/VM or Linux router that is typically used in conjunction with a Guest LAN. Switching logic resides in the z/VM Control Program (CP) which owns the OSA-Express connection and performs all data transfers between Guest LAN nodes and the OSA-Express. This eliminates the overhead associated with a router running in a virtual machine performing this same function.

Additionally, the VSWITCH can run in a high availability configuration. This is done by providing multiple OSA cards and multiple z/VM TCPIP stacks controller. Recovery of either a failing OSA Express card or a failing TCP/IP controller is transparent and automatic to the Linux guests. A block diagram of a VSWITCH with failover is shown in Figure 1-3 on page 4.

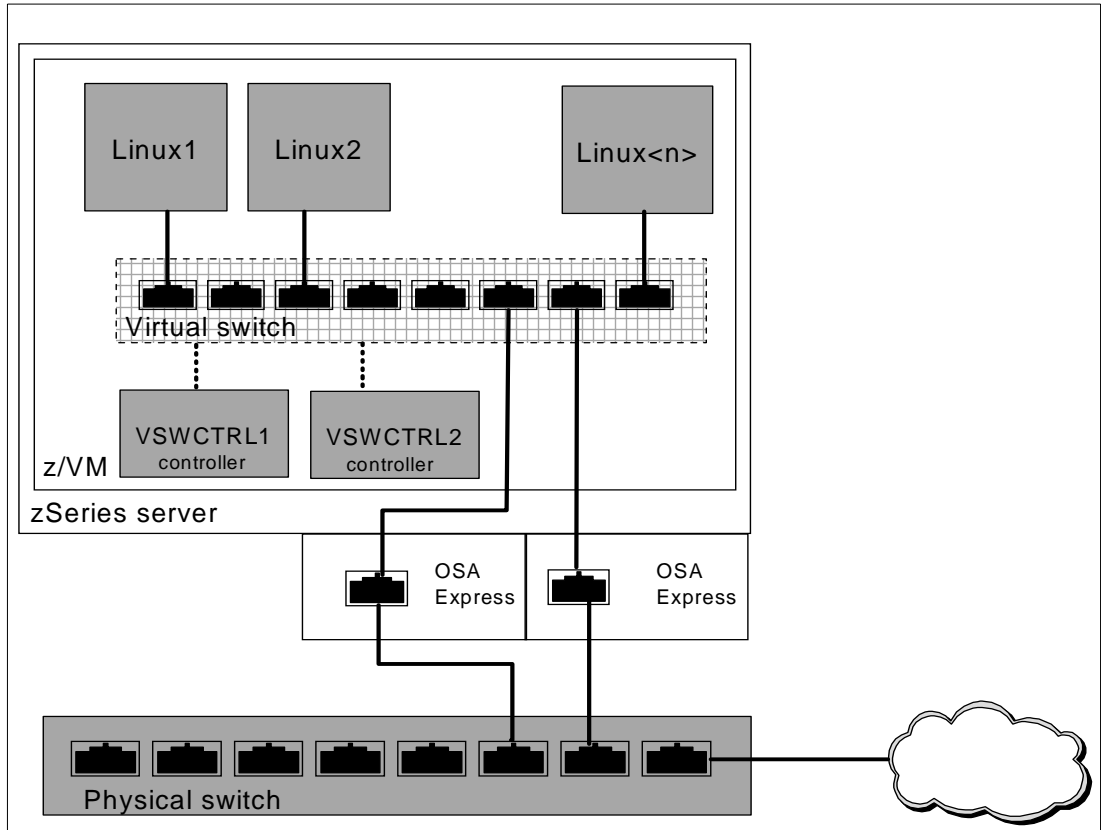


Figure 1-3 VSWITCH with failover block diagram

## 1.2 Setting up VSWITCH with failover

In the following sections a z/VM VSWITCH is set up in high availability mode, to demonstrate its resilience. The examples shown were performed on a “vanilla” z/VM 4.4 system

### 1.2.1 Steps involved

Please refer to Chapter 7 in the *z/VM Virtual Machine Operation* manual, or the *TCP/IP Planning and Customization* manual, for detailed information on the commands used.

The high level steps in the process are in the sections that follow:

- ▶ “Create two new VSWITCH controllers” on page 5
- ▶ “Customize the TCPMAINT user ID” on page 7
- ▶ “Define the VSWITCH” on page 8
- ▶ “Activate the TCP/IP controllers” on page 10
- ▶ “Grant Linux Access to the VSWITCH” on page 11
- ▶ “Shutdown and re-IPL z/VM” on page 12
- ▶ “Attach a Linux TCP/IP stack on LINUX1” on page 13

## Create two new VSWITCH controllers

The user ID TCP/IP is not modified as this is typically used for production z/VM network access. Two new user IDs are created to become VSWITCH controllers. In this example they are named VSWCTRL1 and VSWCTRL2.

To do this, two new entries must first be added to the user directory. The example below shows how this is done via the USER DIRECT file. If you are using a different directory maintenance product such as DirMaint or VM:Directory, the steps will be different, but the result should be the same.

The TCP/IP virtual machine definition is copied twice. Following is the existing definition:

```
USER TCP/IP TCP/IP 32M 128M ABG
  INCLUDE TCPCMSU
  OPTION QUICKDSP SVMSTAT MAXCONN 1024 DIAG98 APPLMON
  SHARE RELATIVE 3000
  IUCV ALLOW
  IUCV ANY PRIORITY
  IUCV *CCS PRIORITY MSGLIMIT 255
  IUCV *VSWITCH MSGLIMIT 65535
  LINK 4TCP/IP40 491 491 RR
  LINK 4TCP/IP40 492 492 RR
  LINK TCPMAINT 591 591 RR
  LINK TCPMAINT 592 592 RR
  LINK TCPMAINT 198 198 RR
  MDISK 191 3390 2294 005 440W01 MR RTCP/IP WTCP/IP MTCP/IP
*
```

For each of the two new user IDs, only two statements need to be modified:

- ▶ New user IDs and passwords are set.
- ▶ A new 191 disk of 5 cylinders is allocated from the end of the 440W02 volume.

XEDIT the file USER DIRECT C and add the two new user ID definitions. After the changes are made the definitions are as follow:

```
USER VSWCTRL1 VSWCTRL1 32M 128M ABG
  INCLUDE TCPCMSU
  OPTION QUICKDSP SVMSTAT MAXCONN 1024 DIAG98 APPLMON
  SHARE RELATIVE 3000
  IUCV ALLOW
  IUCV ANY PRIORITY
  IUCV *CCS PRIORITY MSGLIMIT 255
  IUCV *VSWITCH MSGLIMIT 65535
  LINK 4TCP/IP40 491 491 RR
  LINK 4TCP/IP40 492 492 RR
  LINK TCPMAINT 591 591 RR
  LINK TCPMAINT 592 592 RR
  LINK TCPMAINT 198 198 RR
  MDISK 191 3390 3328 005 440W02 MR RTCP/IP WTCP/IP MTCP/IP
*
USER VSWCTRL2 VSWCTRL2 32M 128M ABG
  INCLUDE TCPCMSU
  OPTION QUICKDSP SVMSTAT MAXCONN 1024 DIAG98 APPLMON
  SHARE RELATIVE 3000
  IUCV ALLOW
  IUCV ANY PRIORITY
  IUCV *CCS PRIORITY MSGLIMIT 255
  IUCV *VSWITCH MSGLIMIT 65535
  LINK 4TCP/IP40 491 491 RR
```

```

LINK 4TCPIP40 492 492 RR
LINK TCPMAINT 591 591 RR
LINK TCPMAINT 592 592 RR
LINK TCPMAINT 198 198 RR
MDISK 191 3390 3333 005 440W02 MR RTCPIP WTCPIP MTCPIP

```

Save your changes with the XEDIT **FILE** subcommand. The command **DISKMAP USER** is run to test for gaps or overlaps.

```

===> diskmap user
The minidisks with the END option specified in this directory will not be included in the following DISKMAP file.

```

File USER DISKMAP A has been created.

Edit the output file, USER DISKMAP, looking for overlaps and gaps. In this example there is a gap of 1872 cylinders in the 440W02 user volume as you placed the two new 191 disks at the end:

```

===> x user diskmap
...
VOLUME  USERID      CUU  DEVTYPE  START    END    SIZE
440W02  $ALLOCS$    A03   3390    00000   00000  00001
...
                                1456   3327   1872   GAP
                                VSWCTRL1  191   3390   03328  03332  00005
                                VSWCTRL2  191   3390   03333  03337  00005

```

Quit out of that file via **F3** and bring the changes online via the **DIRECTXA** command:

```

===> directxa user
z/VM USER DIRECTORY CREATION PROGRAM - VERSION 4 RELEASE 4.0
EOJ DIRECTORY UPDATED AND ON LINE

```

Logon to each of the new user IDs. The 191 disks are new so they will need to be initialized via the **FORMAT** command. For example:

```

USERID  ===> vswctr11
PASSWORD ===>
...
z/VM Version 4 Release 4.0, Service Level 0000 (64-bit),
built on IBM Virtualization Technology
There is no logmsg data
FILES:  NO RDR,  NO PRT,  NO PUN
LOGON AT 10:46:04 EDT TUESDAY 04/27/04
z/VM V4.4.0    2003-06-18 08:53

```

```

DMSACP112S A(191) device error
===> format 191 a
DMSFOR603R FORMAT will erase all files on disk A(191). Do you wish to continue?
Enter 1 (YES) or 0 (NO).
yes
DMSFOR605R Enter disk label:
VSW191
DMSFOR733I Formatting disk A
DMSFOR732I 5 cylinders formatted on A(191)

```

Link and access TCP/IP's 191 disk via the **VMLINK** command and copy all files to the new controller's 191 disk via the **COPYFILE** command:

```

===> vmlink tcpip 191
ENTER READ PASSWORD:
RTCPIP
DMSVML2060I TCP/IP 191 linked as 0120 file mode Z

```

Verify that the source (TCPIP 191 - R/O) is your Z disk and the target (TCPIP 191 - R/W) is your target disk:

```

===> q disk
LABEL  VDEV M  STAT  CYL TYPE BLKSZ  FILES  BLKS USED-(%)  BLKS LEFT  BLK TOTAL
VSW191 191  A  R/W   5 3390 4096    0      7-01    893     900
MNT190 190  S  R/O  100 3390 4096   692   14303-79   3697   18000
MNT19E 19E  Y/S R/O  250 3390 4096  1009   26522-59  18478  45000
TCP191 120  Z  R/O   5 3390 4096    4     13-01    887     900

```

Copy all files from your Z disk to your A disk:

```

===> copy * * z = = a

```

Look at the files via the FILEL command. They should look something like the following:

```

===> filel
Cmd  Filename Filetype Fm Format Lrecl  Records  Blocks  Date  Time
PROFILE EXEC     A2 V      72      25      1 6/28/04 14:53:11
PROFILE TCPERROR A1 V      172     108     2 6/28/04 14:53:11
TCPIP  MESSAGE  A1 V      113      2      1 6/28/04 14:53:11
VMSES  PARTCAT  A1 V      81      1      1 6/28/04 14:53:11

```

**LOGOFF** of VSWCTRL1 and logon to VSWCTRL2 and repeat the steps in this section.

The user IDs VSWCTRL1 and VSWCTRL2 are now created, customized and almost ready to become VSWITCH controllers.

## Customize the TCPMAINT user ID

Logon to TCPMAINT. Four new files must be created on the TCPMAINT 198 disk:

SYSTEM DTCPARMS	The configuration file for customizing TCP/IP servers on your system. It will consist of two lines: one for each of the new controllers. ( <b>Note:</b> this file may already exist on your system)
VSWCTRL1 TCPIP	A sample PROFILE TCPIP for the first VSWITCH controller
VSWCTRL2 TCPIP	A sample PROFILE TCPIP for the second VSWITCH controller
VSWCTRL EXEC	An "exit" that is used to copy the previous samples to each VSWITCH controllers as the file PROFILE TCPIP.

If your TCPMAINT user ID does not have a PROFILE XEDIT file, you may wish to first create one to customize how XEDIT behaves. The following example sets the case to mixed (so text is not folded to upper case) and sets the ruler off:

```

===> x profile xedit
/* simple PROFILE XEDIT */
'SET CASE MIXED IGNORE'
'SET SCALE OFF'

```

Create the file VSWCTRL1 TCPIP as follows. This is a TCPIP configuration file that contains the minimal statements needed to create a VSWITCH controller. The most important statement is the last:

```

===> x vswctrl1 tcpip d
NOTRACE ALL
OBEY
  OPERATOR TCPMAINT MAINT
ENDOBEY
VSWITCH CONTROLLER ON

```

Duplicate the previous file so its file name is the user ID of the second controller via the **COPYFILE** command:

```
====> copy vswctr1 tcpip d vswctr12 = =
```

Now create an *exit program* named **VSWCTRL**. An *exit*, typically a REXX EXEC, allows you to receive information about and control a server as its initialization progresses. The file will be **VSWCTRL EXEC D**. This REXX EXEC will copy the TCPIP configuration files you just created to each of the **VSWITCH** controllers' 191 disks as the file **PROFILE TCPIP A**:

```
====> x vswctr1 exec d
/* EXEC to copy <userid> TCPIP to the 191 disk as PROFILE TCPIP */
/* z/VM 5.1 will do this automatically so this exit won't be needed */
arg parm .
if parm <> "SETUP" then exit 0
'COPYFILE' userid() 'TCPIP D PROFILE TCPIP A (REPLACE TYPE'
exit 0
```

z/VM 5.1 has added support to TCP/IP to read the file **<userid> TCPIP** automatically if it exists (before **<nodeid> TCPIP**). Therefore if you are at z/VM 5.1 or when you move to it, you will not need the **:Exit.VSWCTRL** tags in the **SYSTEM DTCPARMS** file nor in the **VSWCTRL EXEC**.

You have now created two TCPIP configuration files, one for each controller. You have created an exit program to copy these files. Now create the file **SYSTEM DTCPARMS** (if it already exists then edit it) to tie it together:

```
====> x system dtcparms d
:nick.VSWCTRL1 :type.server :class.stack :exit.VSWCTRL
:nick.VSWCTRL2 :type.server :class.stack :exit.VSWCTRL
```

This file is read when services are brought up. It defines two servers with the names **VSWCTRL1** and **VSWCTRL2**, of type **stack** with an exit named **VSWCTRL**.

You now have customized z/VM's TCP/IP so both **VSWCTRL1** and **VSWCTRL2** can act as **VSWITCH** controllers.

## Define the VSWITCH

The **VSWITCH** named **VSW1** can be created dynamically with a **CP DEFINE VSWITCH** command. However such a definition would not persist across IPLs. So that the **VSWITCH** always exists it is recommended that the definition be placed in the **SYSTEM CONFIG** file.

To create a **VSWITCH** that allows for failover, two OSA triplets are needed - one triplet for each path to the physical switch. The OSA CHPIDs must be of type **OSD**. Each of these must be on two different OSA CHPIDs. To achieve a higher level of availability, each of these should be on two different OSA features

**LOGOFF** of **TCPMaint** and **LOGON** to **MAINT**. The available free OSA connections can be queried via the following command:

```
====> q osa free
OSA 0800 FREE , OSA 0801 FREE , OSA 0802 FREE , OSA 0A00 FREE
OSA 0A01 FREE , OSA 0A02 FREE
```

In this example, there are two OSA triplets available starting at addresses 0800 and 0A00.

In order to edit the **SYSTEM CONFIG** file, the CP disk **MAINT CF1** must be accessed read/write. It can be queried via the **QUERY CPDISK** command and must first be taken offline via the **CPRELEASE** command:

```
====> q cpdisk
Label Userid Vdev Mode Stat Vol-ID Rdev Type StartLoc EndLoc
```



```

MNTCF1 MAINT   OCF1 A  R/O 440RES 0200 CKD      391      435
MNTCF2 MAINT   OCF2 B  R/O 440RES 0200 CKD      436      480
MNTCF3 MAINT   OCF3 C  R/O 440RES 0200 CKD      481      540
===> cprel a
CPRELEASE request for disk A scheduled.
HCPZAC6730I CPRELEASE request for disk A completed.
===> q cpdisk
Label Userid  Vdev Mode Stat Vol-ID Rdev Type   StartLoc   EndLoc
MNTCF2 MAINT   OCF2 B  R/O 440RES 0200 CKD      436      480
MNTCF3 MAINT   OCF3 C  R/O 440RES 0200 CKD      481      540

```

Note that the CP disk A is no longer accessed. Now the MAINT CF1 disk can be linked read/write and accessed as a conventional minidisk. Then the SYSTEM CONFIG file can be updated:

```

===> link * cf1 cf1 mr
===> acc cf1 f
===> x system config f

```

Add some lines at the bottom of the file to define the VSWITCH VSW1 attaching it to the two OSA triplets. In this example the two triplets start at addresses 800 and A00. You may also want to add a **VMLAN MACPREFIX** statement. This will ensure the first half of all MAC addresses will be the same on this z/VM LPAR.

Unless you have very special requirements, do not specify **CONTROLLER**; allow the switch to select any available controller. A clause such as **PORTNAME PORTA PORTB** is not necessary because the requirement for OSA port names was removed on both the OSA hardware (with microcode Driver 3G - EC stream J11204, MCL032, OSA level 3.33, November 2003), and z/VM 4.4. Therefore the commands to create the VSWITCH and set the MAC address prefixes are the following:

```

/* define vswitch named vsw1 and set MAC address prefixes to 02-00-01 */
define vswitch vsw1 rdev 800 A00
vmlan macprefix 020001

```

Save your work and use the **CPSYNTAX** utility to check for syntax errors in the SYSTEM CONFIG file:

```

===> access 193 g
===> cpsyntax system config f
CONFIGURATION FILE PROCESSING COMPLETE -- NO ERRORS ENCOUNTERED.

```

When the syntax is correct, put the SYSTEM CONFIG file back on-line:

```

===> rel f
===> det cf1
DASD OCF1 DETACHED
===> cpacc * cf1 a
CPACCESS request for mode A scheduled.
HCPZAC6732I CPACCESS request for MAINT's OCF1 in mode A completed.

```

Query the CP disks one more time to verify your changes are back online.

```

===> q cpdisk
Label Userid  Vdev Mode Stat Vol-ID Rdev Type   StartLoc   EndLoc
MNTCF1 MAINT   OCF1 A  R/O 440RES 0200 CKD      391      435
MNTCF2 MAINT   OCF2 B  R/O 440RES 0200 CKD      436      480
MNTCF3 MAINT   OCF3 C  R/O 440RES 0200 CKD      481      540

```

You now have a VSWITCH named VSW1 set to be defined at IPL time. You can shutdown and reIPL now, or you can define the VSWITCH interactively via the same command as in the SYSTEM CONFIG, as is done in this example:

```

====> define vswitch vsw1 rdev 800 A00
VSWITCH SYSTEM VSW1 is created
HCPSWU2832E The connection for VSWITCH SYSTEM VSW1 is not active.
HCPSWU2832E No TCP/IP controller is available.

```

The “No controller” error is expected because the VSWITCH controllers are not yet active. The VSWITCH cannot be used until the controllers are active.

## Activate the TCP/IP controllers

Query the VSWITCH controllers via the **QUERY CONTROLLER** command:

```

====> q contr
HCPSWQ2835E CONTROLLER * does not exist

```

There are currently no controllers because the two VSWITCH controller user IDs are not logged on. You could use the **XAUTOLOG** command to automatically logon the two controllers, however, it is recommended that you logon to at least the first controller interactively to observe the messages.

Logoff of MAINT and logon to VSWCTRL1. A TCP/IP stack begins to come up because you copied the files from TCP/IP’s 191 disk to the two new controllers. It should find the PROFILE TCP/IP configuration file that was copied from the TCPMAINT 198 disk via the VSWCTRL EXEC exit. If the stack comes up cleanly do not logoff but disconnect via the command **#cp disc**:

```

USERID ====> vswctr11
PASSWORD ====>
...
LOGON VSWCTRL1
z/VM Version 4 Release 4.0, Service Level 0000 (64-bit),
built on IBM Virtualization Technology
There is no logmsg data
FILES: NO RDR, NO PRT, NO PUN
LOGON AT 15:26:10 EDT MONDAY 06/28/04
z/VM V4.4.0 2003-06-18 08:53
<Enter>
DMSACP723I D (198) R/0
DMSACP723I E (591) R/0
DMSACP723I F (592) R/0
Ready; T=0.01/0.01 15:26:11
DTCRUN1022I Console log will be sent to user TCPMAINT
DTCRUN1021R To cancel TCP/IP Stack startup, type any non-blank character and
press ENTER. To continue startup, just press ENTER.
<Enter>
DTCRUN1011I Server started at 15:26:12 on 28 Jun 2004 (Monday)
DTCRUN1011I Running "TCP/IP"
DTCTCP001I z/VM TCP/IP Level 440
***** 06/28/04 *****
15:26:12 DTCIPI008I Initializing... TCPIP MODULE E1 dated 06/18/03 at 13:24
15:26:12 DTCDIP010I UnlockAll issuing "CP UNLOCK VSWCTRL1 0 1FFF"
Command complete
15:26:12 DTCIPI009I LCS devices will use diagnose 98 real channel program support
15:26:12 DTCIPI012I TCP/IP running under z/VM system
15:26:12 DTCIPI005I Trying to open ZVMV4R40 TCPIP *
15:26:12 DTCIPI005I Trying to open PROFILE TCPIP *
15:26:12 DTCIPI006I Using profile file PROFILE TCPIP *
15:26:12 DTCFPM004I FPSM is available with 0 pages allocated
15:26:12 DTCFPM005I FPSM may allocate an additional 2751 pages out of 3057
15:26:12 DTCIUC014I IUCV initializing:
15:26:12 DTCPRI385I Device VSWITCHDEV:
15:26:12 DTCPRI386I Type: VSWITCH-IUCV, Status: Not started

```

```

15:26:12 DTCPRI387I      Envelope queue size: 0
15:26:12 DTCPRI389I      VM id: *VSWITCH
15:26:12 DTCIUC031I      TCP/IP server connected to *VSWITCH system service
...
15:26:12 DTCOSD080I      VSWITCH-OSD initializing:
15:26:12 DTCPRI385I      Device VSW10800DEV:
15:26:12 DTCPRI386I      Type: VSWITCH-OSD, Status: Not started
15:26:12 DTCPRI387I      Envelope queue size: 0
15:26:12 DTCPRI388I      Address: 0800
15:26:12 DTCQDI001I      QDIO device VSW10800DEV device number 0802:
15:26:12 DTCQDI007I      Enabled for QDIO data transfers
15:26:12 DTCOSD238I      ToOsd: Multicast support enabled for VSW10800DEV
15:26:12 DTCOSD319I      ProcessSetArpCache: Supported for device VSW10800DEV
15:26:12 DTCOSD341I      Obtained MAC address 020000000009 for device VSW10800DEV
==>#cp disc

```

Do the same for the second VSWITCH controller, VSWCTRL2. You should not see as many informational messages as this stack does not become a controller.

When both stacks are up you have successfully created and activated a primary and backup VSWITCH controller.

## Grant Linux Access to the VSWITCH

Disconnect from VSWCTRL2 if you haven't already. Logon to AUTOLOG1. The following command will grant LINUX1 access to the VSWITCH named VSW1:

```
==> SET VSWITCH VSW1 GRANT LINUX1
```

However, this is a interactive command so LINUX1 will only have access to VSW1 for the life of this IPL. For LINUX1 to have access to the VSWITCH at z/VM IPL time, the **SET VSWITCH** command is added to the AUTOLOG1 user ID (**Note**: this function can also be performed in the SYSTEM CONFIG file, but the statement is **MODIFY VSWITCH**, not **SET VSWITCH** see 1.4.3, "MODIFY VSWITCH statement and SET VSWITCH command" on page 18 for more details).

AUTOLOG1 is the user ID that z/VM uses to issue commands and start services each time z/VM is booted (for those of you with z/OS experience this is like using the COMMANDxx members in sys1.parm1ib. For UNIX/Linux experienced people, this is like adding scripts in the /etc/init.d/ run level infrastructure).

The file PROFILE EXEC on the user AUTOLOG1 is modified to grant LINUX1 access to VSW1. Also the two VSWITCH controllers are started by including an **XAUTOLOG** command for TCP/IP. Refer to the *CP Commands and Utilities* manual for more details on **XAUTOLOG**.

```

==> file1
Cmd  Filename Filetype Fm Format Lrecl  Records  Blocks  Date    Time
x    PROFILE  EXEC    A2 V      42       6        1 11/07/94 9:06:57

```

If the file has never been modified, you should see something like the following:

```

/*****/
/* Autolog1 Profile Exec */
/*****/
ADDRESS COMMAND CP AUTOLOG VMSERVS VMSERVS
ADDRESS COMMAND CP AUTOLOG VMSERVU VMSERVU
ADDRESS COMMAND CP AUTOLOG VMSERVR VMSERVR

```

The existing three lines can be deleted if you are not using the z/VM shared file system. Modify the PROFILE EXEC as follows:

```

/*****/
/* Autolog1 Profile Exec */

```

```

/*****/
ADDRESS COMMAND 'CP XAUTOLOG TCPIP'
ADDRESS COMMAND 'CP XAUTOLOG VSWCTRL1'
ADDRESS COMMAND 'CP XAUTOLOG VSWCTRL2'
ADDRESS COMMAND 'CP SET VSWITCH VSW1 GRANT LINUX1'

```

(Note: the first command, **XAUTOLOG TCPIP**, is not required for the VSWITCH controllers, though this is typically the first virtual machine brought up for the production TCP/IP stack)

The order in which you start the TCPIP stacks will determine which user ID will be the primary VSWITCH controller. In this example it will be VSWCTRL1.

## Shutdown and re-IPL z/VM

To test the setup, z/VM is shutdown and re-IPLed (Note: be sure you are in a position to do this and be sure you know how to get to the console if you are accessing z/VM via TCP/IP and the stack does not come up):

```

===> shutdown reipl
...

```

When the system is finished re-IPLing, logon to MAINT and see which virtual machines came up via the **QUERY NAMES** command. Look for the machines you specified in AUTOLOG1's PROFILE EXEC:

```

===> q n
VSWCTRL2 - DSC , VSWCTRL1 - DSC , TCPIP - DSC , OPERSYMP - DSC
AUTOLOG1 - DSC , DISKACNT - DSC , EREP - DSC , OPERATOR - DSC
MAINT - 0009
VSM - TCPIP

```

Now query the VSWITCH:

```

===> q vswitch
VSWITCH SYSTEM VSW1      Type: VSWITCH  Active: 0      MAXCONN: INFINITE
  PERSISTENT RESTRICTED  NONROUTER  MFS: 8192     ACCOUNTING: OFF
  State: Ready
  CONTROLLER: VSWCTRL1   IPTIMEOUT: 5      QUEUESTORAGE: 8
  PORTNAME: UNASSIGNED   RDEV: 0800 VDEV: 0800
  PORTNAME: UNASSIGNED   RDEV: 0A00

```

This shows that the VSWITCH VSW1 has been created using the virtual device starting at address 800, with a backup device starting at address A00. Now query the controllers:

```

===> q contr
CONTROLLER VSWCTRL1 Available: YES  VDEV Range: *
  SYSTEM VSW1      Controller: *
CONTROLLER VSWCTRL2 Available: YES  VDEV Range: *

```

This shows that the VSWCTRL1 user ID is the primary controller and that the user ID VSWCTRL2 is available as a backup. Now check the access to the VSWITCH:

```

===> q vswitch access
VSWITCH SYSTEM VSW1      Type: VSWITCH  Active: 0      MAXCONN: INFINITE
  PERSISTENT RESTRICTED  NONROUTER  MFS: 8192     ACCOUNTING: OFF
  State: Ready
  CONTROLLER: VSWCTRL1   IPTIMEOUT: 5      QUEUESTORAGE: 8
  PORTNAME: UNASSIGNED   RDEV: 0800 VDEV: 0800
  PORTNAME: UNASSIGNED   RDEV: 0A00
  Authorized userids:
    LINUX1  VLAN: ANY
    SYSTEM  VLAN: ANY

```

This shows that LINUX1 has access to the VSWITCH. You have now tested your VSWITCH with failover setup across IPLs. You are now ready to connect Linux images.

### Attach a Linux TCP/IP stack on LINUX1

In this example, the Linux user ID LINUX1 will be given a NIC at virtual addresses 600-602. It is recommended that a USER DIRECT entry create the virtual NIC. This will make it more persistent and avoids the necessity of a **COUPLE** command to attach Linux to the VSWITCH. A virtual NIC can be created via either the **SPECIAL** or **NICDEF** statements in the USER DIRECTORY, however, the **NICDEF** statement is recommended. The statement can be put in each Linux user directory entry, or just once in a USER DIRECT profile:

```
nicdef 600 type qdio lan system VSW1
```

See 1.4.1, "The user directory NICDEF statement" on page 16 for more details.

For completeness, the Linux profile and LINUX1 user definition are as follow. You will probably choose to define your Linux images differently:

```
PROFILE LINDFLT
  ip1 cms
  mach esa 4
  iucv any
  iucv allow
  cpu 00 base
  cpu 01
  nicdef 600 type qdio lan system VSW1
  SPOOL 000C 2540 READER *
  SPOOL 000D 2540 PUNCH A
  SPOOL 000E 1403 A
  CONSOLE 009 3215 T
  LINK MAINT 0190 0190 RR
  LINK MAINT 019D 019D RR
  LINK MAINT 019E 019E RR
  link tcpmaint 592 592 rr
...
user linux1 linux1 256M 512M G
  include lindflt
  mdisk 100 3390 0001 0100 zvmpr1 mr read write multi
  mdisk 101 3390 0101 3238 zvmpr1 mr read write multi
  mdisk 102 3390 0001 3138 zvmpr2 mr read write multi
  mdisk 103 3390 3139 0100 zvmpr2 mr read write multi
  mdisk 191 3390 3239 0100 zvmpr2 mr read write multi
```

The file is saved and the changes are brought on line:

```
===> directxa user
z/VM USER DIRECTORY CREATION PROGRAM - VERSION 4 RELEASE 4.0
EOJ DIRECTORY UPDATED AND ON LINE
```

Now the user ID LINUX1 has a virtual NIC at addresses 600-602 which is automatically coupled to the VSWITCH upon logon. When you first logon to the Linux ID, you should see the NIC being created:

```
LOGON LINUX1
NIC 0600 is created; devices 0600-0602 defined
```

You can get more information about the NIC from the CP **QUERY NIC** command:

```
===> q nic
00: CP Q NIC
00: Adapter 0600 Type: QDIO Name: UNASSIGNED Devices: 3
```

```
00: Port 0 MAC: 02-00-01-00-00-00 VSWITCH: SYSTEM VSW1 MFS: 8192
```

Note that the name is unassigned and the MAC address prefix is 02-00-01. As CP assigns MAC address suffixes, they will count up from 00-00-00.

In this example, a Linux distribution is installed in the LINUX1 user ID with a TCP/IP address of 129.40.35.90 using the NIC at addresses 600-602. Details of this are beyond the scope of this paper. A Linux SSH session is started and a few items are worth pointing out: the device addresses in the file `/etc/chandev.conf`, the association of the qeth driver with the eth0 interface in `/etc/modules.conf` and the output of the `ifconfig eth0` command:

```
# tail -1 /etc/chandev.conf
noauto;qeth0,0x0600,0x0601,0x0602;add_parms,0x10,0x0600,0x0602,portname:suse1in7
# grep eth0 /etc/modules.conf
alias eth0 qeth
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 02:00:00:00:00:00
          inet addr:129.40.35.90  Mask:255.255.255.0
          inet6 addr: fe80::200:0:100:0/10 Scope:Link
          UP RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:99 errors:0 dropped:0 overruns:0 frame:0
          TX packets:84 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:5873 (5.7 Kb)  TX bytes:10476 (10.2 Kb)
          Interrupt:3
```

Also, you can query the NIC with the `CP QUERY NIC` command. Note that the name is now the default port name that the SuSE Linux install process assigns and that the MAC address suffix has been incremented:

```
===>#CP Q NIC
Adapter 0600 Type: QDIO      Name: suse1in7  Devices: 3
Port 0 MAC: 02-00-01-00-00-01 VSWITCH: SYSTEM VSW1  MFS: 8192
```

A Linux image is now connected to the VSWITCH and available for some tests.

## 1.3 Test the VSWITCH

Now that your highly available VSWITCH is built, you will want to perform a few tests to see how it works. You will want to test losing an OSA device and losing a VSWITCH controller.

### 1.3.1 Losing an OSA device

The loss of an OSA device is simulated via the `DETACH` command. First look at the status of the VSWITCH:

```
===> q vswitch
VSWITCH SYSTEM VSW1      Type: VSWITCH  Active: 1      MAXCONN: INFINITE
PERSISTENT RESTRICTED  NONROUTER  MFS: 8192     ACCOUNTING: OFF
State: Ready
CONTROLLER: TCP/IP      IPTIMEOUT: 5      QUEUESTORAGE: 8
PORTNAME: UNASSIGNED    RDEV: 0800 VDEV: 0800
PORTNAME: UNASSIGNED    RDEV: 0A00
```

Note that the primary controller is TCP/IP and the active OSA triplet begins at address 0800. Switch to a Linux session and start a `ping` that goes through the VSWITCH and the gateway:

```
# ping 129.40.45.11
PING 129.40.45.11 (129.40.45.11) from 129.40.35.91 : 56(84) bytes of data.
```

```

64 bytes from 129.40.45.11: icmp_seq=1 ttl=58 time=1.38 ms
64 bytes from 129.40.45.11: icmp_seq=2 ttl=58 time=1.40 ms
64 bytes from 129.40.45.11: icmp_seq=3 ttl=58 time=1.41 ms
64 bytes from 129.40.45.11: icmp_seq=4 ttl=58 time=1.63 ms
64 bytes from 129.40.45.11: icmp_seq=5 ttl=58 time=1.42 ms
64 bytes from 129.40.45.11: icmp_seq=6 ttl=58 time=1.47 ms
64 bytes from 129.40.45.11: icmp_seq=7 ttl=58 time=1.57 ms
64 bytes from 129.40.45.11: icmp_seq=8 ttl=58 time=1.49 ms
...

```

While the ping is running on Linux, go over to a z/VM session and detach the active OSA triplet to simulate an OSA failure. You should see that z/VM will attach the secondary OSA triplet:

```

===> q 0800-0802 0a00-0a02
OSA 0800 ATTACHED TO VSWCTRL1 0800
OSA 0801 ATTACHED TO VSWCTRL1 0801
OSA 0802 ATTACHED TO VSWCTRL1 0802
OSA 0A00 FREE , OSA 0A01 FREE , OSA 0A02 FREE
===> det 800-802 tcpip
0800-0802 DETACHED TCPIP

```

Watch the ping output on the Linux session. As you detach the primary OSA from the VSWITCH controller, you should see the ping output continue without dropping a single packet. If this happens, z/VM has successfully failed over to the second OSA device. Verify this with the **QUERY VSWITCH** command. Note that device A00 is now active:

```

===> q 0800-0802 0a00-0a02
OSA 0800 FREE , OSA 0801 FREE , OSA 0802 FREE
OSA 0A00 ATTACHED TO VSWCTRL1 0A00
OSA 0A01 ATTACHED TO VSWCTRL1 0A01
OSA 0A02 ATTACHED TO VSWCTRL1 0A02

```

Including the **DETAILS** parameter allows you to see that the virtual device begins at A00, and that no packets were lost or in error.

```

===> q vswitch det
VSWITCH SYSTEM VSW1      Type: VSWITCH  Active: 1      MAXCONN: INFINITE
  PERSISTENT RESTRICTED  NONROUTER  MFS: 8192     ACCOUNTING: OFF
  State: Ready
  CONTROLLER: VSWCTRL1   IPTIMEOUT: 5      QUEUESTORAGE: 8
  PORTNAME: UNASSIGNED   RDEV: 0800
  PORTNAME: UNASSIGNED   RDEV: 0A00 VDEV: 0A00
  RX Packets: 30          Discarded: 0      Errors: 0
  TX Packets: 30          Discarded: 0      Errors: 0
  RX Bytes: 1836         TX Bytes: 3080
  Authorized userids:
    LINUX1  VLAN: ANY
    SYSTEM  VLAN: ANY
...

```

You can also detach the secondary OSA triplet and note that z/VM will attach the primary back to the VSWITCH controller, if available:

```

===> det 0a00-0a02 vswctr11
0A00-0A02 DETACHED VSWCTRL1
===> q 800-802 a00-a02
OSA 0800 ATTACHED TO VSWCTRL1 0800
OSA 0801 ATTACHED TO VSWCTRL1 0801
OSA 0802 ATTACHED TO VSWCTRL1 0802
OSA 0A00 FREE , OSA 0A01 FREE , OSA 0A02 FREE

```

This test has shown that a z/VM VSWITCH can failover between OSA devices and Linux IDs are not affected.

### 1.3.2 Losing a VSWITCH controller

The other failover scenario is losing a VSWITCH controller. Begin a ping again from Linux to another system. Simulate losing a VSWITCH controller by forcing the current controller off the system.

```
===> force vswctrl1
USER DSC LOGOFF AS VSWCTRL1 USERS = 8 FORCED BY MAINT
```

Again you should see the ping continue without losing a packet. Query the VSWITCH and the controllers again and you should see that VSWCTRL2 is now the controller:

```
===> q vswitch
VSWITCH SYSTEM VSW1 Type: VSWITCH Active: 1 MAXCONN: INFINITE
PERSISTENT RESTRICTED NONROUTER MFS: 8192 ACCOUNTING: OFF
State: Ready
CONTROLLER: VSWCTRL2 IPTIMEOUT: 5 QUEUESTORAGE: 8
PORTNAME: UNASSIGNED RDEV: 0800 VDEV: 0800
PORTNAME: UNASSIGNED RDEV: 0A00
===> q contr
CONTROLLER VSWCTRL2 Available: YES VDEV Range: *
SYSTEM VSW1 Controller: *
```

Logging VSWCTRL1 back on will result in it being available again as a backup controller:

```
===> xautolog vswctrl1
Command accepted
AUTO LOGON *** VSWCTRL1 USERS = 9
HCPCLS6056I XAUTOLOG information for TCP/IP: The IPL command is verified by the I
PL command processor.
===> q contr
CONTROLLER VSWCTRL1 Available: YES VDEV Range: *
CONTROLLER VSWCTRL2 Available: YES VDEV Range: *
SYSTEM VSW1 Controller: *
```

This test has shown that z/VM can failover between VSWITCH controllers and Linux TCP/IP communication is not affected.

## 1.4 Reference

The syntax of some statements and commands is included for convenience. For a complete description, see the z/VM manuals on the Web at:

<http://www.vm.ibm.com/pubs/>

- ▶ *z/VM CP Planning and Administration*, SC24-6043
- ▶ *z/VM CP Command and Utility Reference*, SC24-6008
- ▶ *z/VM TCP/IP Planning and Customization*, SC24-6019

### 1.4.1 The user directory NICDEF statement

The syntax for the user directory NICDEF statement is as follows:

```
NICDEF vdev [TYPE HIPERsockets|QDIO] [LAN owner name]
[DEVICES devs]
[CHPID nn]
```



[MACID nnnnnn]

where:

- ▶ <vdev>
  - Specifies the (hex) base device address where this NIC is installed in a virtual machines I/O configuration
  - A block of contiguous device addresses is allocated to this NIC, beginning with “vdev” (see DEVICES devs)
- ▶ TYPE HIPERsockets or QDIO
  - Must connect (couple) to a Guest LAN of the same type
- ▶ DEVICES <devs>
  - Specifies (decimal) number of I/O devices created as part of this NIC, beginning with “vdev”
  - The default/minimum is 3 devices for Read-Control, Write-Control and Data
- ▶ LAN <owner name>
  - Specifies the Guest LAN or VSWITCH to connect to:
    - “owner” is a VM user ID or SYSTEM
    - “name” is the name specified on the **DEFINE LAN** or **DEFINE VSWITCH** command
- ▶ CHPID <nn>
  - Identifies the (hex) Channel Path ID number to use for this NIC
  - CP automatically assigns an available CHPID to the adapter if you omit the option
  - This option is only required for z/OS guests connecting to a HiperSockets Guest LAN
- ▶ MACID nnnnnn
  - Optionally specify the Medium Access Control (MAC) identifier for this adapter
  - CP automatically generates a unique identifier for the adapter if you omit the option
  - If specified, the six (hex) digits are appended to the MACPREFIX to form a locally administered address

## 1.4.2 DEFINE VSWITCH command

The syntax for the CP **DEFINE VSWITCH** command is as follows:

```
DEFINE VSWITCH switchname [RDEV NONE|rdev]
[CONnect|DISCONnect]
[QUEuestorage 8M|numberM]
[CONTRoller *|userid]
[IPTimeout 5|nnn]
[NONrouter|PRIrouter]
[PORTname portname]
```

where:

- ▶ <switchname>
  - Is a 1-8 character token naming the VSWITCH
  - A VSWITCH is “SYSTEM” owned
- ▶ RDEV <rdev>
  - Identifies the base (hex) device address used to connect the switch to a real QDIO ethernet OSA-Express card
  - Up to three OSA devices can be specified
  - NONE - signifies a standalone switch with no external connection

- ▶ CONnect vs. DISCONnect
  - Indicates whether the device identified by the RDEV operand should be activated
  - CONnect is the default value
- ▶ QUEEstorage <numberM>
  - Indicates the upper limit of fixed storage that can be consumed for QDIO queues
  - 8M is the maximum and default value
- ▶ CONTRoller <userid>
  - Identifies z/VM TCPIP stack that will control the QDIO ethernet OSA-Express card
  - "\*" is the default value, meaning that CP can select from any eligible z/VM TCPIP stack
- ▶ IPTimeout <nnn>
  - Indicates the amount of time that a remote IP address is retained in the VSWITCH IP address table
  - The valid range is 1 - 120 minutes (default is 5 minutes)
- ▶ NONrouter vs. PRIrouter
  - Indicates whether the QDIO ethernet OSA-Express card identified by the RDEV keyword will act as a router to the VSWITCH
  - PRIrouter is only required if a guest connected to the VSWITCH will be serving as a router
- ▶ PORTname <portname>
  - A 1-8 character name that identifies the QDIO OSA-Express device
  - Up to three port names can be specified
  - Multiple port names are used when multiple OSA devices are specified on the RDEV operand

### 1.4.3 MODIFY VSWITCH statement and SET VSWITCH command

Granting Linux user IDs access to VSWITCHes can be done in the SYSTEM CONFIG file or via a CP command. The syntax is slightly different.

#### In the SYSTEM CONFIG file:

```
MODIFY VSWITCH switchname [GRANT userid] [VLAN ANY|vlanid]
```

#### Via a CP command:

```
CP SET VSWITCH switchname [GRANT userid] [VLAN ANY|vlanid]
```

Either the statement or the command can use the same parameters:

```
[REVOKE userid]
[PORTname portname]
[RDEV NONE|rdev]
[CONnect|DISCONnect]
[QUEEstorage numberM]
[CONTRoller *|userid]
[NONrouter|PRIrouter]
```

## 1.5 Thanks

Thanks to Phil Smith III and Dennis Musselwhite for reviewing this paper. Special thanks to Alan Altmark for giving an in-depth review.