Using z/VM FBA Emulation in a Second-Level ID

Vic Cross
Systems and Technology Group
IBM Australia

January 2012

Abstract

The z/VM FBA Emulation feature allows z/VM systems to access and use SCSI-over-FCP disks via a SAN fabric. These disks (often referred to as EDEVs, from the command that creates them) are straightforward to use at a first-level system but complexity arises at second-level. When used with a second-level z/VM system the disks can be accessed in two ways: either by using the FBA Emulation feature of the second-level z/VM system, or as full-pack minidisks from the first-level system. This article presents these alternatives, and the benefits of using one or the other method.

Contents

Conventions

Technical papers usually use character formatting and other techniques to convey additional meaning, and this one is no exception. Here are the typographical conventions used in this paper:

Bold
: Used for program names, as well as commands and their options.

Italic
: In commands and listings, italics signifies arguments or variables that should be replaced with values supplied by the reader. Italics are also used for URLs, new terms, file names and extensions, directories, and comments.

`Typewriter`
: Used to show output from commands, or the contents of files.

`Bold Typewriter`
: In examples, this shows commands or other input that should be literally typed by the reader.

`Italic Typewriter`
: In examples, this shows input (arguments, variables) that should be replaced with values supplied by the reader.

This kind of text box provides some additional information to help you understand the topic.

This kind of text box draws your attention to an important aspect of the topic being discussed.

If you see one of these text boxes, it contains a vital piece of information that might help you avoid errors or downtime in your configuration. Ignore these at your own risk!

Revision History

Version	Date	Author	Notes
1.0	12 January 2012	Vic Cross	First edition
0.1	October 2011	Vic Cross	Draft (unpublished)

1 Review of FBA Emulation and EDEVICEs

z/VM can access SCSI disks through the services of an I/O layer known as FBA Emulation, introduced with z/VM 5.1. FBA Emulation was added to z/VM to remove one of the "barriers to entry" for Linux on System z: ECKD DASD. With FBA Emulation, a site with no traditional mainframe installation can install and run z/VM and Linux guests without having to invest in mainframe DASD.

The FBA Emulation layer depends on normal I/O to operate, because it uses FCP ports presented to the z/VM LPAR to attach to the SAN. Because of the fact that some I/O to a SCSI LUN passes through CP twice (once as I/O to the EDEV, then again as I/O to the FCP subchannel) there is some performance loss when using FBA Emulation compared to native SAN access. However, FBA Emulation can still provide I/O rates that far exceed the requirements of most installations, and some FBA Emulation paths (such as paging to an EDEVICE) have been heavily optimised to reduce overhead. In addition, EDEVICEs offer many disk management benefits for virtual machines that require access to SAN disks (rather than having the virtual machines access the SAN directly).

SCSI disks accessed using FBA Emulation are often referred to as EDEVICEs or EDEVs, after the commands used to define them. For each SCSI disk accessed using an EDEVICE, an emulated Fixed Block Architecture (FBA) disk (model 9336) is presented to CP.

It might be tempting to refer to this as a *virtual disk*, but it's not a virtual device. In z/VM we use the term *virtual disk* to describe an instance of one of the disk virtualisation technologies, like minidisks and VDISKs (virtual-disks-in-storage). Emulated disk is a better term for the SCSI support, since the real SCSI LUN from the SAN is represented block-for-block by the EDEVICE, with no virtualisation involved.

1.1 Defining an EDEVICE

The way to define a disk using FBA Emulation is either with the CP **SET EDEVICE** command or in *SYSTEM CONFIG* using the **EDEVICE** statement. The syntax of the SET EDEVICE command is shown in Appendix A. The syntax for the **EDEVICE** statement in *SYSTEM CONFIG* is similar, shown in Appendix B.

If you use HCD/HCM to manage your z/VM I/O configuration, you can add EDEVICEs to your IODF. That way, your logical definitions for SCSI disks can be managed in the same place as your hardware definitions. Refer to the z/VM manuals for more information about using HCD/HCM to manage your I/O configuration.

Once a SCSI LUN is defined as an EDEVICE, you can use it as if it was a Model 9336 FBA DASD. The CP **QUERY EDEVICE** command lets you find out the status of your EDEVICEs, as shown in Listing 1.

```
Q EDEVICE ALL
EDEV 5C00 TYPE FBA ATTRIBUTES 2145
EDEV 5C01 TYPE FBA ATTRIBUTES 2145
EDEV 5C02 TYPE FBA ATTRIBUTES 2145
EDEV 5C03 TYPE FBA ATTRIBUTES 2145
EDEV 5C04 TYPE FBA ATTRIBUTES 2145
EDEV 5C11 TYPE FBA ATTRIBUTES 2145
EDEV 5C12 TYPE FBA ATTRIBUTES 2145
EDEV 5C13 TYPE FBA ATTRIBUTES 2145
EDEV 5C14 TYPE FBA ATTRIBUTES 2145
EDEV 5C15 TYPE FBA ATTRIBUTES 2145
EDEV 5C16 TYPE FBA ATTRIBUTES 2145
```

```
Ready; T=0.01/0.01 10:42:24
Q EDEVICE 5C11
EDEV 5C11 TYPE FBA ATTRIBUTES 2145
Ready; T=0.01/0.01 10:42:59
Q EDEVICE 5C11 DETAILS
EDEV 5C00 TYPE FBA ATTRIBUTES 2145
  VENDOR: IBM PRODUCT: 2145 REVISION: 0000
  BLOCKSIZE: 512 NUMBER OF BLOCKS: 41943040
  PATHS:
    FCP_DEV: 1000 WWPN: 50050768014011E0 LUN: 0065000000000000 NOTPREF
      CONNECTION TYPE: SWITCHED
    FCP_DEV: 1100 WWPN: 5005076801301207 LUN: 0065000000000000 PREF
      CONNECTION TYPE: SWITCHED
    FCP_DEV: 1400 WWPN: 5005076801101207 LUN: 0065000000000000 PREF
      CONNECTION TYPE: SWITCHED
    FCP_DEV: 1500 WWPN: 50050768012011E0 LUN: 0065000000000000 NOTPREF
      CONNECTION TYPE: SWITCHED
    FCP_DEV: 1000 WWPN: 5005076801401207 LUN: 0065000000000000 PREF
      CONNECTION TYPE: SWITCHED
    FCP_DEV: 1100 WWPN: 50050768013011E0 LUN: 0065000000000000 NOTPREF
      CONNECTION TYPE: SWITCHED
    FCP_DEV: 1400 WWPN: 50050768011011E0 LUN: 0065000000000000 NOTPREF
      CONNECTION TYPE: SWITCHED
    FCP_DEV: 1500 WWPN: 5005076801201207 LUN: 0065000000000000 PREF
      CONNECTION TYPE: SWITCHED
Ready; T=0.01/0.01 10:43:04
```

Listing 1: CP QUERY EDEVICE examples

When an EDEVICE is defined using CP **SET EDEVICE**, it is initially offline (EDEVICEs defined in *SYSTEM CONFIG* or in the IODF are brought online to the system automatically). Path validation occurs when the device is brought online. Listing 2 shows an example of defining an EDEVICE where there is a problem preventing the system from reaching the LUN.

```
1  set edev 5C51 type fba attributes 2145 FCP_DEV 1000 wwpn 5005076801 4011e0 lun ↩
     →020a000000000000
2  EDEV 5C51 was created.
3  Ready; T=0.01/0.02 10:53:36
4  set edev 5C51 type fba attributes 2145 add path FCP_DEV 1100 wwpn 5005076801301207 ↩
     →lun 020a000000000000
5  EDEV 5C51 was modified.
6  Ready; T=0.01/0.01 10:54:01
7  q edevice 5c51 details
8  EDEV 5C51 TYPE FBA ATTRIBUTES 2145
9    PATHS:
10     FCP_DEV: 1000 WWPN: 50050768014011E0 LUN: 020A000000000000 NOTPREF
11     FCP_DEV: 1100 WWPN: 5005076801301207 LUN: 020A000000000000 NOTPREF
12 Ready; T=0.01/0.01 10:54:09
13 vary online 5c51
14 HCPSZP8701I Path FCP_DEV 1000 WWPN 50050768014011E0 LUN 020A000000000000 was deleted ↩
     →from EDEV 5C51 because it is invalid.
15 HCPSZP8701I Path FCP_DEV 1100 WWPN 5005076801301207 LUN 020A000000000000 was deleted ↩
     →from EDEV 5C51 because it is invalid.
16 HCPCPN8700I Emulated Device 5C51 cannot be varied online because
17 HCPCPN8700I there are no valid paths defined to the device.
18 1 device(s) specified; 0 device(s) successfully varied online
19 Ready(08700); T=0.01/0.01 10:54:13
20 q edevice 5c51 details
21 EDEV 5C51 TYPE FBA ATTRIBUTES 2145
22   PATHS:
23     No paths exist.
24 Ready; T=0.01/0.01 10:54:32
25 set edev 5C51 type fba attributes 2145 add path FCP_DEV 1000 wwpn 5005076801 4011e0 ↩
     →lun 0071000000000000
26 EDEV 5C51 was modified.
27 Ready; T=0.01/0.01 11:38:39
28 set edev 5C51 type fba attributes 2145 add path FCP_DEV 1100 wwpn 5005076801301207 ↩
     →lun 0071000000000000
29 EDEV 5C51 was modified.
30 Ready; T=0.01/0.01 11:38:50
31 q edev 5c51 details
32 EDEV 5C51 TYPE FBA ATTRIBUTES 2145
```

```
33    PATHS:
34      FCP_DEV: 1000 WWPN: 50050768014011E0 LUN: 0071000000000000 PREF
35      FCP_DEV: 1100 WWPN: 5005076801301207 LUN: 0071000000000000 NOTPREF
36  Ready; T=0.01/0.01 11:38:55
37  vary online 5c51
38  5C51 varied online
39  1 device(s) specified; 1 device(s) successfully varied online
40  Ready; T=0.01/0.01 11:39:07
41  q edev 5c51 details
42  EDEV 5C51 TYPE FBA ATTRIBUTES 2145
43    VENDOR: IBM PRODUCT: 2145 REVISION: 0000
44    BLOCKSIZE: 512 NUMBER OF BLOCKS: 41943040
45    PATHS:
46      FCP_DEV: 1000 WWPN: 50050768014011E0 LUN: 0071000000000000 PREF
47        CONNECTION TYPE: SWITCHED
48      FCP_DEV: 1100 WWPN: 5005076801301207 LUN: 0071000000000000 NOTPREF
49        CONNECTION TYPE: SWITCHED
50  Ready; T=0.01/0.01 11:39:34
```

Listing 2: Defining an EDEVICE, with problems and resolution

- At line 1 the EDEVICE and the first path are defined. The command on line 4 adds a second path to the disk.
- The output at line 8 shows that the EDEVICE is defined and has two paths. However, no information about the device is available because CP has not yet established a connection to the LUN.
- CP is asked to bring the device online at line 13. FBA Emulation tries to establish a connection to the storage. In this case, both the paths to the device have failed validation – this might be because the storage device on the SAN is unreachable, or because the storage device is not allowing access to the LUN from our host.
- With no valid paths to the device, FBA Emulation reports that it cannot vary the device online. The messages from line 16 indicate this.
- Line 21 shows the output of another query command on the EDEVICE. Because FBA Emulation deletes invalid paths from an EDEV, the definition reports "No paths exist". To get this device working, it will be necessary to resolve the SAN issue and then run **SET EDEVICE** commands with the **ADD PATH** keywords to add valid paths back to the EDEVICE definition.
- After verifying details with our storage team, it is found that the number being used to identify the LUN was incorrect. At lines 25 and 28 the correct path details are added. This time, the **VARY ONLINE** command succeeds (shown on line 38).
- To check the new EDEVICE, **QUERY EDEVICE DETAILS** is run again. The output is shown from line 42. Now that CP has established connection to the LUN, details such as blocksize and block count are shown.

To further illustrate how the SCSI LUN is represented as if it were an actual DASD, the output of some of the "traditional" DASD management query commands is shown in Listing 3.

```
1   q 5c51
2   DASD 5C51 V1DA13
3   Ready; T=0.01/0.01 11:41:58
4   q 5c51 status
5   DEVICE 5C51 OPERATIONAL
6   Ready; T=0.01/0.01 11:42:04
7   q 5c51 id
8   DASD 5C51 9336-10 CU: 6310-80
9   Ready; T=0.01/0.01 11:42:08
10  q da v1da13
```

```
11 DASD 5C51 V1DA13
12 Ready; T=0.01/0.01 11:46:11
13 q da details 5c51
14 5C51  CUTYPE = 6310-80, DEVTYPE = 9336-10, VOLSER = V1DA13, CYLS = 53980
15       BLKS = 41943040
16 Ready; T=0.01/0.01 11:47:09
```

Listing 3: Traditional CP QUERY commands on an EDEVICE

1.2 What makes EDEVICEs different?

The way that FBA Emulation presents disks to CP makes SCSI disks different to traditional disks like ECKD[1]. With ECKD, disks are actually present in the system I/O definition as unit addresses. With FBA Emulation, only the FCP devices used to access the SAN are present in the system I/O definition, and the EDEVICEs are created separately.

The second important difference between traditional DASDs and EDEVICEs, the aspect that we will be investigating in this article, is the way the devices are accessed from a second-level ID. For DASDs, the access method used at second-level is the same as first-level: SSCH-based I/O to the DASD. Whether the DASD is a minidisk or a dedicated device, the access method is the same. This is not the case for EDEVICEs, as we see in the next section.

2 Presenting EDEVICEs to a second-level system

The difference in the way that EDEVICEs are created and managed leads to an important difference when using EDEVICEs created on a first-level z/VM system with a z/VM system running at second-level. You actually have a choice for how to get access to these devices from the second-level system:

- Use full-pack minidisks hosted on the first-level EDEVICEs, which the second-level system will see as "real" 9336 disks;
- Use EDEVICEs within the second-level system, using the FBA Emulation capability in the second-level system as it would be used first-level.

In most cases the first option is the easiest, but it does mean that if the system is later IPLed first-level that a different process is used to IPL first-level versus second-level. The EDEVICE definitions that would be required to connect to the SCSI disks will not be present at second-level and therefore would not be tested. While it is somewhat harder to use second-level EDEVICEs, you do have the advantage that your testing is more representative of how the system will operate when it runs first-level.

Full-pack or standard sub-pack minidisks can be used for disks assigned to a second-level z/VM system. However, if the second-level system needs to be IPLable at first-level, full-pack minidisks are required. If full-pack minidisks are not used, the second-level system can only be IPLed second-level (unless the disks are copied to full-pack minidisks). Giving the second-level z/VM system access to real cylinder 0 (or block 0 for FBA) is a pre-requisite for being able to later IPL the second-level system at first-level. This requirement applies to all disk types, both DASDs (ECKD or FBA) and SCSI EDEVs.

[1] Traditional disks here refers to ECKD disks such as 3390, but could also include real FBA disks if you have such devices.

SCSI disk access scenarios are shown in Figure 1. The diagram shows a first-level z/VM system, and a user in that z/VM system called ZVMCLONE in which another z/VM system runs. The disks for both the first- and second-level systems are SCSI disks accessed using FBA Emulation.

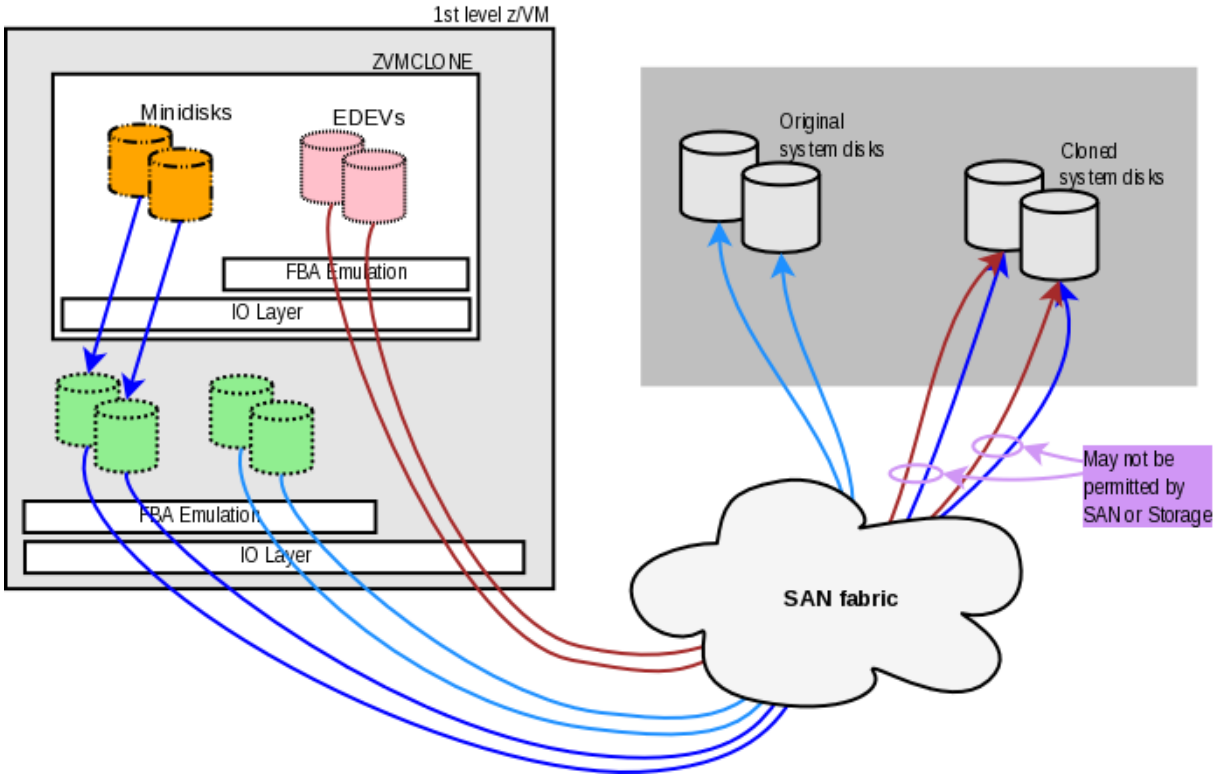


Figure 1: Accessing SCSI disks from second-level user

The green disks represent the EDEVICEs defined at the first-level system for both its own disks (the "Original system disks") and the cloned system disks. Connectivity for these disks flows through the FBA Emulation layer in CP, then through the I/O subsystem and the SAN fabric to the storage subsystem.

When we look at the second-level system, the orange disks represent the use of MDISK statements in the ZVMCLONE user to present devices to the z/VM system running in ZVMCLONE. The blue lines representing this disk access do not pass through the FBA Emulation layer in the second-level CP. However as we have discussed, it is possible to use the FBA Emulation layer in the second-level CP to connect to the SAN and represent the SCSI disks as EDEVICEs, as shown by the pink disks.

At the storage subsystem end, this is a second logical connection from the System z CPC to the same LUNs already accessed at the first-level system. This may not be permitted by the SAN switches or by the storage device.

In the second-level z/VM system in Figure 1 the orange disks and pink disks actually refer to the same LUNs in the storage subsystem, but because they are accessed using different methods z/VM treats them as different disks. This is where confusion can arise: CP will see duplicate volume labels, and the actual path to the disk may not be the path that you intended.

3 Using second-level EDEVICEs

Using EDEVICEs at second-level is the best way to do the most accurate test of how your cloned z/VM system will behave at first-level. You need to be aware that your SAN or storage subsystem may prevent you from accessing the disk multiple times from the same host, so you will need to do one of the following to resolve this:

- Remove the EDEVICEs from the first-level system;
- Use different FCP ports to access the disks from the second-level system;
- Use N-Port ID Virtualisation (NPIV) to virtualise the connections from first- and second-level z/VM systems.

You will also have to pay close attention to SAN zoning and storage LUN configuration to make sure that all the appropriate LUNs are accessible.

Even after doing these things, it is possible that you will have to make changes between your test at second-level and your real first-level installation. For example, if you use NPIV the Fibre Channel Worldwide Port Names (WWPNs) will not be the same between the second-level system in one LPAR and the first-level destination in another LPAR. Making changes between the test and installation can undermine the reasoning for using EDEVICEs at second-level.

3.1 Preparing for second-level EDEVICEs

If you want to use the FBA Emulation feature of a second-level z/VM system, the only preparation needed is to give the second-level ID access to FCP subchannels through which it can access the SAN. Use the **DEDICATE** directory statement, or the CP **ATTACH** command, to assign one or more FCP devices to the second-level ID you will use. The way to use **ATTACH** to do this is similar to using DEDICATE; the usage of DEDICATE is shown here.

3.1.1 Recap of DEDICATE

In this context, the syntax of DEDICATE is **DEDICATE** *vdev rdev*, where *vdev* is the virtual device number to be seen by the guest, and *rdev* is the real device number on the host. Listing 4 shows an example of attaching two FCP subchannels to two different virtual machines.

```
USER VMCLONE1 VMCLONE1 64M 1024M G
 ...
 DEDICATE 1000 100A
 DEDICATE 1400 140A
 ...
*
USER VMCLONE2 VMCLONE2 64M 1024M G
 ...
 DEDICATE 1000 100B
 DEDICATE 1400 140B
 ...
```

Listing 4: DEDICATE directory control statement example

The standard rules for attaching real devices to a virtual machine apply, most importantly that the device is not in use either by another virtual machine or by CP. The example at Listing 4 shows a scenario where 100A and 140A are the next unallocated device addresses on the two FCP ports being used, so 100A and 140A were allocated to VMCLONE1 and 100B and 140B were allocated to VMCLONE2. The *vdev* parameter only has scope within the guest and does not have to align with

other virtual machines (or the real *rdev*); this is shown above as both VMCLONE1 and VMCLONE2 having FCP devices presented at 1000 and 1400. You can use this to create configurations that are more similar for the operating system running within the guest.

If you plan on moving the second-level z/VM system to first level, you could simplify the later change by making the virtual device number in the **DEDICATE** or **ATTACH** the same as the real device number (if known) in the LPAR the system will run in. This will allow you to define **SET EDEVICE** configurations in your second-level system that won't change when the system moves to first-level (well, the FCP_DEV parameter won't change at least).

3.2 Using z/VM guest SCSI IPL support

If you use second-level EDEVs, you will have to IPL using the SCSI boot loader. The **SET LOADDEV** command is used to specify the FCP WWPN, SCSI LUN ID and boot loader details in preparation for the IPL command. To perform the IPL, provide the device address of an FCP subchannel through which the LUN specified in the SET LOADDEV command can be reached.

Listing 5 shows the commands used to IPL from a SCSI disk with LUN number 0123000000000000 at a storage device with FCP WWPN 50050768014011E0 using FCP port 1010. The z/VM IPL will bring up the stand-alone program loader (SAPL) on the terminal at device address 2300.

```
1 SET LOADDEV CLEAR PORT 50050768 014011E0 LUN 01230000 00000000 BR_LBA C8
2 IPL 1010 LOADPARM 2300
```

Listing 5: IPL using guest SCSI IPL support

The SCSI IPL information can be set in the directory entry for a user using the **LOADDEV** directory control statement. The syntax is the same as for the **SET LOADDEV** command, except that the CLEAR keyword is not used and the hexadecimal values in the PORT and LUN parameters do not have the space separator after the eighth digit. If the SCSI IPL data is specified in the directory, there is no need to use the SET LOADDEV command before the IPL command (unless the data in the LOADDEV directory control statment needs to be changed).

4 Using second-level minidisks

If you use minidisks to present your EDEVICEs to the second-level user, some tasks will be simpler. If your second-level z/VM system will always run that way (i.e. it will never be IPLed first-level in its own LPAR) it's much easier to use minidisks:

- No need to define EDEVs in the second-level system;
- No need to allocate FCP subchannels to the second-level system (and co-ordinate between possible multiple second-level IDs needing FCP devices);
- No problems with SAN zoning or LUN masking (the first-level system is the only FCP host the SAN sees);
- "Normal" DASD IPL process, instead of the more complex SCSI boot loader IPL process.

There may be other aspects that may be more complex, however. If you are planning to move your second-level system to first-level at a later time, you will need to make many changes to the second-level system configuration. EDEVICE definitions will only appear at the first-level system, but the possibility that you will have to make SAN zoning changes between your second-level test and first-level installation is still present.

4.1 Moving second-level to first-level

The biggest issue to be aware of is that before you move the system to first-level you will have to update *SYSTEM CONFIG* in the cloned system for all the EDEVICEs you will need, and by using minidisks those *SYSTEM CONFIG* changes will be untested. In addition, if you use minidisks and IPL the system second-level with EDEVICE definitions in place, there will be a large number of error messages generated by the FBA Emulation feature as it tries to bring up EDEVICEs that are not accessible. You can ignore the messages, but they may be distracting and might divert attention from a legitimate error condition. An example of the kind of messages seen in this situation is shown in Listing 6.

```
1  17:23:44 HCPSZP8701I Path FCP_DEV 1000 WWPN 50050768014011E0 LUN 0018000000000000 was ↩
     →deleted from EDEV 5C03 because FCP device 1000 does not exist.
2  17:23:44 HCPSZP8701I Path FCP_DEV 1100 WWPN 5005076801301207 LUN 0018000000000000 was ↩
     →deleted from EDEV 5C03 because FCP device 1100 does not exist.
3  17:23:44 HCPSZP8701I Path FCP_DEV 1400 WWPN 5005076801101207 LUN 0018000000000000 was ↩
     →deleted from EDEV 5C03 because FCP device 1400 does not exist.
4  17:23:44 HCPSZP8701I Path FCP_DEV 1500 WWPN 50050768012011E0 LUN 0018000000000000 was ↩
     →deleted from EDEV 5C03 because FCP device 1500 does not exist.
5  17:23:44 HCPIIO8700I Emulated Device 5C03 cannot be varied online because
6  17:23:44 HCPIIO8700I there are no valid paths defined to the device.
```

Listing 6: Error messages from failed EDEVICE startup

5 Choosing your method: minidisks or EDEVICEs

There is no right or wrong way to choose – as we have seen, each method has advantages and disadvantages. Depending on the potential role of the z/VM system you're running second-level however, there is a slight bias:

- If your second-level z/VM will always run second-level, minidisks are the easier approach.
- If the second-level z/VM is going to be run first-level, second-level EDEVs provides a more consistent approach between running first- and second-level.

The choices you have could be summarised as "avoiding confusion" or "managing confusion".

5.1 Avoiding confusion

One possible recommendation is to choose one way or the other – do not define both full-pack minidisks and EDEVICEs for accessing your disks, as there is a risk of confusion and possible system corruption. The easiest way to handle this is to avoid the confusion and decide to use only one type or the other. If you use full-pack minidisks, then ensure that you don't use EDEVICEs at all in the second-level system. Alternatively, if you use second-level EDEVICEs, don't define any full-pack minidisks to the second-level ID.

If you have full-pack minidisk definitions, one way to prevent disk conflicts would be to not give your second-level ID any FCP devices (either with **DEDICATE** statements or the **ATTACH** command). This would make sure that, even if you have EDEVICE statements in your second-level *SYSTEM CONFIG*, the second-level system will not have access to an FCP subchannel to be able to connect to the SAN. Although this would certainly eliminate the possibility of confusion between EDEVICEs and full-pack minidisks, some flexibility in managing the configuration is lost.

5.2 Managing confusion

One of the compelling reasons for choosing z/VM to host Linux workloads is the flexibility available in choosing the best way to host each individual workload in your environment. While choosing one or another method will simplify things, you may end up making part of your system operate less efficiently.

Take the following scenario as an example: you have used the full-pack minidisk method to clone a new z/VM system which will run at first-level. To prepare the system to go live you need to configure userids and install some software; the software will be installed on a new SAN disk which will be dedicated to the new system. Since that disk is not going to be used by the first-level z/VM, it would be easier to define that disk directly to the second-level system using an EDEVICE rather than use a full-pack minidisk. You can do this safely even though you decided to use full-pack minidisks for the clone of z/VM – mixing different attachments for *different disks* has little risk of confusion – and defining the disk only to the second-level system avoids some unneccesary effort on the first-level system.

"Managing confusion" will give you the best flexibility to configure disks as needed, in ways that make sense and are effective for your installation. As we'll see in the next section, taking control of the disk configuration will allow some very sophisticated configurations to be built.

5.3 Synchronised virtual device addresses

For a system which you want to work using either native EDEVs or first-level full pack minidisks, a tip to make some operations on the system easier is to use the *same device numbers* for the minidisk definitions in the first-level ID as the EDEVICE statements in the second-level *SYSTEM CONFIG*. This will cause the disks to appear at the same addresses regardless of which method is used, which means that any system configuration that depends on the disk device address (such as a minidisk definition using the DEVNO parameter) would work in either scenario. Also, the PDVOL IPL parameter necessary when IPLing from a SCSI volume would be the same, allowing a "normal" IPL to be performed every time (instead of having to use SAPL to set PDVOL at each IPL, **SALIPL** can be used to store a consistent PDVOL value in the IPL text).

This technique, used with caution, can even provide limited protection against both access methods being available simultaneously. With synchronised virtual device addresses, when z/VM tries to define the EDEVs contained in *SYSTEM CONFIG* it would find existing devices – the minidisks from first-level – at those device addresses. CP would be unable to define the EDEVs, but the same disks are already being provided as virtual disks so the system still gets access to the disks it needs. Because of this, the technique does give preference to minidisk access since a pre-existing minidisk will always cause the EDEVICE definition to fail. Also, as explained before, the inability to define the EDEVs would cause IPL error messages to appear, and these messages would have to be disregarded when performing a minidisk IPL.

For all the potential elegance that this method has at the outset, there is potential for confusion. For example, in the situation above if the intention was to IPL from EDEVs, the SCSI boot loader IPL process would be used to IPL the guest. With the minidisk devices present, at best the IPL might fail with a strange and confusing code, but at worst it might successfully IPL! If the EDEV error messages were missed (or disregarded), an administrator would likely believe that since a SCSI IPL was done that the system was actually running using its own EDEVs, and might erroneously remove the LUNs from the first-level system.

5.4 Use record modifiers to streamline the process

Whether using synchronised addresses as described above or a different method, you could use record modifiers in your *SYSTEM CONFIG* file to further tune your IPL process. The exact details are left as an exercise for the reader (cards and letters please!) but one example might be using the system identifier to disable EDEVICE statements for disks that are obtained via full-pack minidisks when the system is running in a second-level ID.

6 Conclusion

SCSI disk access using z/VM's FBA Emulation feature is straightforward for first-level z/VM systems, but there are a number of considerations with disk access for second-level systems. FBA Emulation can be used in a second-level z/VM system the same as in a first-level system, which means that z/VM administrators have a choice about how to give disk resources to a second-level z/VM system. There is no right or wrong way, but this article covered many of the issues that administrators need to consider when making the decision that is best for their installation.

6.1 About the author

Vic Cross is a Client Technical Specialist with IBM's Systems and Technology Group. Vic has been involved in Linux on the mainframe since the days the mainframe was called "System/390". He has worked on many Linux on System z implementations in Australia and New Zealand, and is an ITSO Platinum Author (most recently having worked on "Security for Linux on System z", which you can download at www.redbooks.ibm.com/abstracts/sg247728.html). He also presents at conferences such as the IBM System z Technical Symposium, and on the ITSO System z World Tour. Vic and his family live in Brisbane, Australia.

You can contact Vic about this paper at viccross@au1.ibm.com.

6.2 Acknowledgements

Thank you to the following individuals, who helped in the development of this paper:

Eric Farman
IBM z/VM Development
Endicott, NY, USA

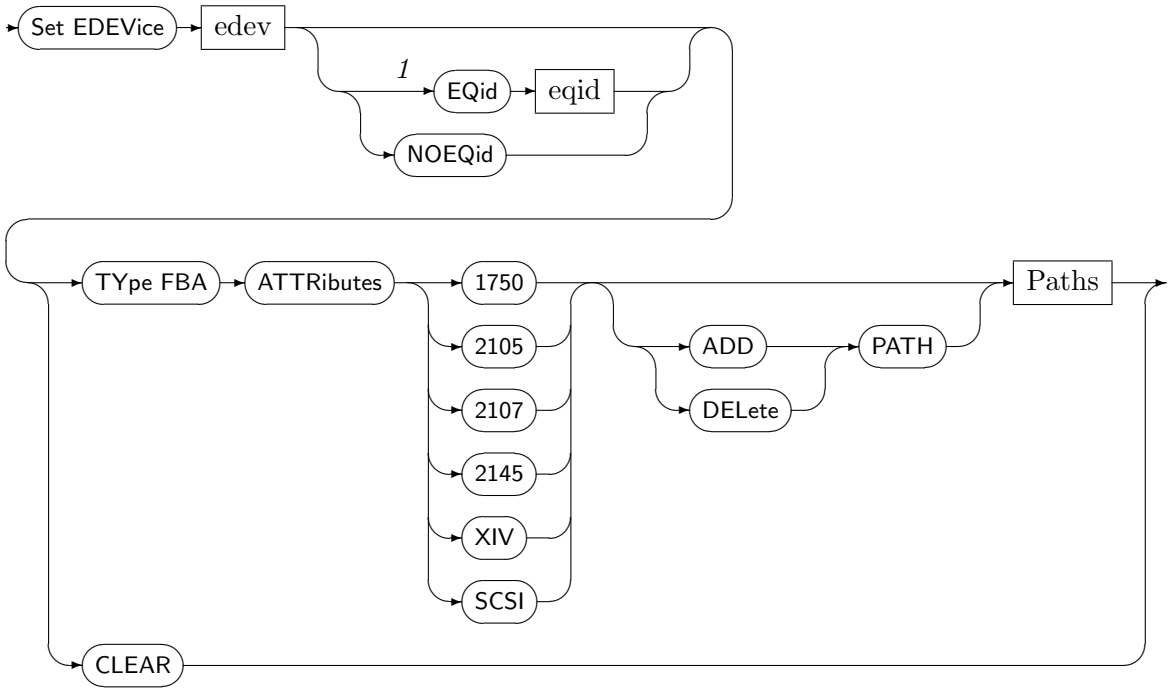
Rod Nash
SUSE
Melbourne, Australia

Tim O'Sullivan and **David Gray**
IBM Systems and Technology Group
Brisbane, Australia

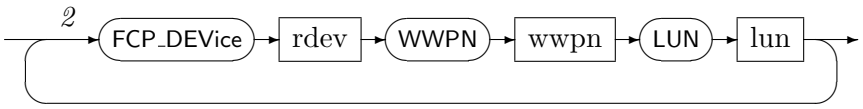
A CP SET EDEVICE command

The syntax of CP SET EDEVICE from z/VM 6.2 is shown here. Earlier z/VM releases may be different from what is shown here, or may support different options (for example, the EQID and NOEQID options were introduced in z/VM 6.2; EDIQs were introduced with z/VM 6.2 as part of support for SSI).

CP SET EDEVICE



Paths



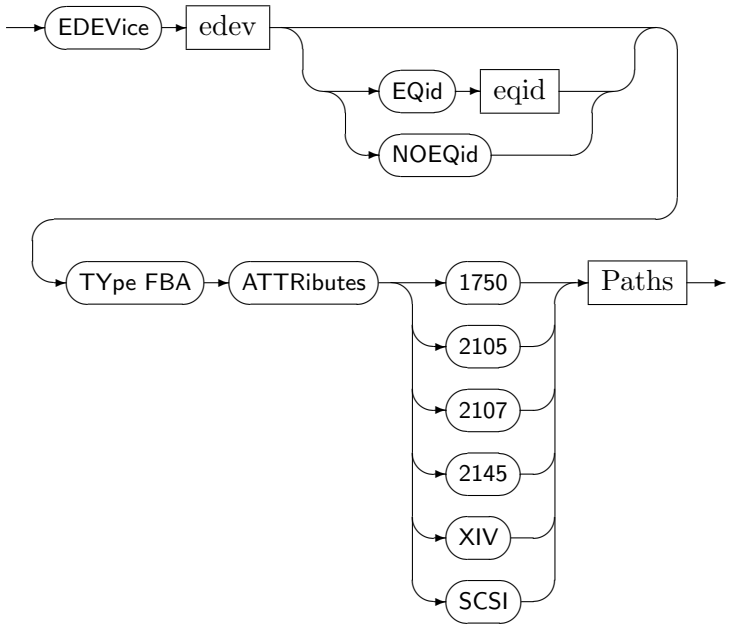
Notes:

1. Normally, system-generated EQIDs will be assigned to EDEVICEs. The ability to manually assign EQIDs is provided as a means of assigning device EQIDs in the unexpected event that system generation of an EDEVICE EQID fails.
2. You can specify a maximum of 8 paths to the device.

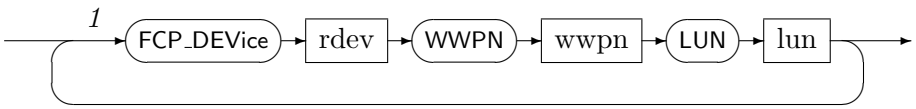
B EDEVICE statment

The syntax of the *SYSTEM CONFIG* **EDEVICE** statement from z/VM 6.2 is shown here. Earlier z/VM releases may be different from what is shown here, or may support different options (for example, the EQID and NOEQID options were introduced in z/VM 6.2; EDIQs were introduced with z/VM 6.2 as part of support for SSI).

EDEVICE



Paths



Notes:

1. You can specify a maximum of 8 paths to the device.