Redpaper

Alan Kline
John Kapernick
Romney White
Reed Mullen

Multiple z/OS Virtual Machines on z/VM

Introduction

This IBM® Redpaper describes some of the possible ways to configure a z/VM® system and a set of z/OS® virtual machines for use in testing z/OS-based tools and products. By supplying each tester with their own virtual machine, one tester has little or no impact on others.

This material is based on 15 years of experience in operating a z/OS test environment on z/VM for developing and testing z/OS middleware in IBM. Our current environment includes three z/VM LPARs on two different processors, hosting over 300 z/OS images and 8 Linux® images. While we have a large DASD pool, we maintain a set of releases dating back to OS/390® Release 10 for service needs, plus a large development environment. As a result, we have implemented an environment where we share as much DASD and as many datasets as possible, while providing sufficient private space for individual development and test needs and a reasonable amount of protection against shared DASD being altered accidentally.

Note: The following examples make reference to z/OS Releases 8 and 9, but are applicable to any release of z/OS.

z/VM

Our basic z/VM environment described in this section includes the following assumptions:

- z/VM Directory Maintenance Facility (DIRMAINT) is used to maintain virtual machine definitions. For more information about DIRMAINT administration, refer to the IBM publication *z/VM Directory Maintenance Facility Tailoring and Administration Guide,* SC24-6135; for information regarding DIRMAINT commands, please refer to the IBM publication *z/VM Directory Maintenance Facility Commands Reference,* SC24-6133.
- RACF/VM might or might not be installed; the following description assumes it is not.
- If the z/OS virtual machines require TCP/IP connectivity, it could be provided through use of real or virtualized network devices. For more information regarding connectivity options, refer to the IBM publication *z/VM Connectivity,* SC24-6080. For the environment

described here, connectivity is provided by virtualized devices: vNICs (Network Interface Card) and virtual VSWITCHes (VSWITCH).

- z/VM allows z/OS virtual machines to be combined into a sysplex if so desired, and provides complete simulation of coupling facility resources and structures to support such configurations.

This environment makes use of z/VM's SYSTEM CONFIG file to provide information about the z/VM installation, as well as a USER DIRECTORY to describe all the virtual machines allowed to use the z/VM system. Much information about SYSTEM CONFIG and the User Directory is available in the IBM publication *z/VM CP Planning And Administration,* SC24-6083.

The environment, outlined here, assumes virtualized connectivity is provided. In particular, VSWITCH technology is employed to allow virtual machines to access the corporate intranet if necessary. VSWITCHes are defined in the SYSTEM CONFIG file. Here is a sample VSWITCH definition (sample file ZVM.SYSTEM.CONFIG in the compressed file described in Appendix B, "Additional material" on page 23).

Example 1 Sample VSWITCH definition

```
Define VSWITCH VSW47 RDEV 1504 CONNECT CONTROLLER * ,
               VLAN 1 PORTTYPE ACCESS PORT OSAFE
   Modify VSWITCH VSW47 GRANT DCEDDE2 VLAN 541
   Modify VSWITCH VSW47 GRANT ZOSAO VLAN 529
```

In Example 1, the name of the VSWITCH is VSW47; it uses a real OSA Express Adapter at device addresses 1504-1506 to connect to the corporate network; it can be controlled by the z/VM TCP/IP stack (CONTROLLER *); it is enabled for VLAN technology and uses a portname of OSAFE. In addition, two virtual machines – DCEDDE2 and ZOSAO – are authorized to access this VSWITCH; DCEDDE2 is configured to use VLAN 541 and ZOSAO is configured to use VLAN 529.

Each virtual machine definition described is created initially using a prototype directory entry referred to by DIRMAINT as a `protodir`. Example 2 is a sample PROTODIR that can be used for z/OS instance creation (sample file ZOSMODEL.PROTODIR as described in Appendix B, "Additional material" on page 23).

Example 2 Sample prototype directory

```
1 USER zosmodel NEW1PASS 512M 1G G
2 INCLUDE COMMVST
3 ACCOUNT 2 ZOSTEST
4 CPU 00 BASE
5 IPL CMS PARM AUTOCR
6 CONSOLE 0680 3215 T OPERATOR
7 NICDEF CF00 TYPE QDIO DEVICES 3 LAN SYSTEM VSW47 CHPID FC
```

In Example 2, virtual machines created with this prototype directory entry would exhibit the following characteristics:

1. Default storage size of 512 MB, with a maximum size of 1 GB; default VM privilege class of G for GENERAL USER.
2. A special DIRMAINT profile called COMMVST will be added to the directory entry to make it easier to maintain the same information in multiple virtual machine definitions.
3. A pair of default ACCOUNT values are established.

4. The virtual machine will have one CPU.
5. The virtual machine will automatically IPL CMS (z/VM's Conversational Monitor System) when it is logged on; it is presumed that additional tools would then be invoked to IPL the proper release of z/OS in this virtual machine.
6. The virtual console for this userid will be at virtual address 0680, of default type 3215, and its z/VM-based output (as opposed to z/OS console messages) will be sent to the z/VM operator if this user gets disconnected from VM somehow.
7. The virtual machine will use a virtual NIC adapter at virtual addresses CF00-CF02 for TCP/IP connectivity; the vNIC will be a QDIO-type device connected to the system VSWITCH VSW47 using CHPID FC, which will be specified in your z/OS HCD as TYPE OSD for QDIO support. You might need to find a channel path that is not configured in your real machine to use for this value.

To create a new virtual machine instance using this **protodir** statement, the z/VM system programmer would issue the command **DIRM ADD**, which returns a panel similar to that shown in Figure 1.

```
---------------------------DirMaint ADD-----------------------------

 Add an entry to the directory for a new USERID or Profile.

 Fill in the USERID or PROFILE being added:
1         ===> ZOSAO

 Optionally fill in the following when using a prototype:
2    LIKE ===> ZOSMODEL (file name of prototype)
3    PW   ===> password (password for new user)
4    VPW  ===> password (password again for verification)
     ACCT ===>          (account value for new user - optional)

 Notes:
    - If a value is given for any one of PW, VPW, or ACCT,
      then a value is required for LIKE.
    - If a value is given for either PW or VPW,
      then a value is required for both of them.


5741-A05 (c) Copyright IBM Corporation 1979, 2007.
 1= Help   2= Prefix Operands    3= Quit 5=Submit 12=Cursor
====>
                                              Macro-read 1 File
```

Figure 1 Creating a new virtual machine instance using the protodir statement

In Figure 1:

1. This field contains the name of the virtual machine instance being created (ZOSAO in the example in this document).
2. This field contains the name of the PROTODIR file to use as the model for the new userid being created (ZOSMODEL PROTODIR in this example).
3. and 4. The PW and VPW fields contain the initial password to be assigned to the new userid.

Once the fields are filled in properly, pressing PF5 creates the new virtual machine instance. The directory entry created with the settings in Figure 1 is shown in Example 3.

Example 3 The directory entry created from the PROTODIR

```
USER ZOSAO NEW1PASS 512M 1G G
INCLUDE COMMVST
ACCOUNT 2 ZOSTEST
CPU 00 BASE
IPL CMS PARM AUTOCR
CONSOLE 0680 3215 T OPERATOR
NICDEF CF00 TYPE QDIO DEVICES 3 LAN SYSTEM VSW47 CHPID FC
```

In this example, the INCLUDE COMMVST statement specifies a profile to be used by this virtual machine. It is used to obtain access to all the shared minidisks needed for operation of this instance. This allows the various instances to share the DASD pool, and the pool directory statements are all maintained in a single place.

Example 4 shows a subset of the statements in the specified common profile COMMVST PROFILE (sample file COMMVST.DIRECT in the compressed file described in Appendix B, “Additional material” on page 23).

Example 4 A subset of statements found in COMMVST PROFILE

```
PROFILE COMMVST
IPL CMS PARM AUTOCR
LOGONBY ADMIN1 ADMIN2
OPTION TODENABLE MAINTCCW
SPECIAL 0500 CTCA RSCS
SPOOL 000C 2540 READER *
SPOOL 000D 2540 PUNCH A
SPOOL 000E 1403 A
LINK MAINT 0190 0190 RR
LINK MAINT 019D 019D RR
LINK MAINT 019E 019E RR
LINK MVSDASD 0191 0191 RR
LINK MVSDASD 0140 0140 MW
LINK MVSDASD 0141 0141 MW
LINK MVSDASD 014D 014D MW
LINK MVSDASD 0C60 0C60 MW
LINK MVSDASD 0C61 0C61 MW
```

In Example 4, the image will IPL CMS by default. The LOGONBY statement, which in this example is configured to allow a couple of system programmers to log on as this userid by specifying their own VM password rather than the password of the z/OS virtual machine, allows for easier support and diagnosis of problems by the support team. The OPTION statement specifies that the virtual machine will be able to change its virtual clock value, and be able to issue diagnostic CCWs to resolve I/O device problems. The SPECIAL statement defines a virtual channel-to-channel adapter (vCTCA) designated for connection to an RSCS virtual machine to provide RSCS-to-NJE connectivity. If you do not plan to use RSCS networking, you can omit this statement. The SPOOL statements define virtual card reader, virtual card punch, and virtual printer that the virtual machine could make use of, if desired. Finally, several DASD minidisks are connected to the virtual machine through the various LINK statements. User MVSDASD owns all the z/OS minidisks defined in the system; various profiles can be defined to fit different needs, such as providing links for only those devices required to meet that need.

Example 5 shows an example of the directory entry for MVSDASD (sample file MVSDASD.DIRECT in the compressed file mentioned in Appendix B, “Additional material” on page 23).

Example 5 MVSDASD directory entry

```
USER MVSDASD NOLOG
MDISK 0191 3390 00001 00020 MDISK1 RRV READPW WRITEPW MULTPW
MDISK 0801 3390 00000 03339 ZDR181 RRV READPW WRITEPW MULTPW
MDISK 0802 3390 00000 03339 ZDR182 RRV READPW WRITEPW MULTPW
MDISK 0803 3390 00000 03339 ZDR183 RRV READPW WRITEPW MULTPW
MDISK 0901 3390 00000 10017 ZDR191 RRV READPW WRITEPW MULTPW
MDISK 0C60 3390 00000 03339 SMBRS1 RRV READPW WRITEPW MULTPW
MDISK 0C61 3390 00000 03339 SMBRS2 RRV READPW WRITEPW MULTPW
MDISK 014D 3390 00001 00834 PAGEV1 RRV READPW WRITEPW MULTPW
MDISK 0140 3390 00000 10017 IMAG00 MWV READPW WRITEPW MULTPW
MDISK 0141 3390 00000 03339 IMAG01 MWV READPW WRITEPW MULTPW
```

Note that in Example 5, z/OS releases 8 and 9 are supported, with z/OS R8 using three volumes of 3390-3 DASD (volumes ZDR181, ZDR182 and ZDR183) and z/OS R9 using one volume of 3390-9 DASD (volume ZDR191). In addition, volumes SMBRS1 and SMBRS2 are defined to be SMS-managed volumes shared by all the z/OS virtual machines, PAGEV1 is defined as a less-than-full-volume minidisk intended to hold a page dataset for an individual z/OS virtual machine; multiple instances of such a minidisk can reside on a single 3390 DASD volume. Volumes IMAG00 and IMAG01 are defined to hold configuration information for all the z/OS virtual machines – for example, shared PARMLIBs and PROCLIBs. One other item to point out in Example 5 is the minidisk defined at virtual device 0191. This is a CMS-formatted minidisk shared by all z/OS test images and set up to hold whatever EXECs are needed to be able to IPL the desired release of z/OS in the virtual machine. Examples of these EXECs are shown at the end of this section.

If the virtual machine needs to be in a sysplex with a virtual coupling facility, then its directory entry will need to include the statement:

```
OPTION CFUSER
```

In addition, the sysplex definition will require two specialized userids: the Coupling Facility Virtual Machine (CFVM) and the Secondary Console for the CFVM. Example 6 is a prototype directory entry for the CFVM (sample file CFCCFVM.DIRECT in the additional material described in Appendix B on page 23).

Example 6 Prototype directory entry for the CFVM

```
USER cfccfvm  password  256M 512M G
CPU 00 BASE
CPU 01
OPTION CFVM
SHARE RELATIVE 1000
XAUTOLOG concfvm
CONSOLE 009 3270 T concfvm
```

Note in the prototype directory entry that `cfccfvm` should be replaced with the userid for your coupling facility virtual machine, and `concfvm` should be replaced with the userid you will use as the secondary console for the coupling facility virtual machine.

Example 7 is a prototype directory entry for the secondary console (sample file CONCFVM.PROTODIR in the additional material described in Appendix B on page 23).

Example 7 Prototype directory entry for the secondary console

```
USER concfvm  SUcfvm  8M 32M G
INCLUDE SECCON
```

Example 8 shows the SECCON profile included in the secondary console virtual machine (sample file SECCON.DIRECT in the additional material described in Appendix B on page 23):

Example 8 SECCON profile

```
PROFILE SECCON
IPL CMS PARM AUTOCR
LOGONBY ADMIN1 ADMIN2
CONSOLE 0009 3215 T
SPOOL 000C 2540 READER *
SPOOL 000D 2540 PUNCH A
SPOOL 000E 1403 A
LINK CFCC 0199 0191 RR
LINK TCPMAINT 0592 0592 RR
LINK MAINT 0190 0190 RR
LINK MAINT 019D 019D RR
LINK MAINT 019E 019E RR
```

Using DIRM ADD to create new userid CFCZOSAO, which will use CFCCFVM PROTODIR, and new userid CONZOSAO, which will use CONCFVM PROTODIR, would create new directory entries as shown in Example 9.

Example 9 Using DIRM ADD to create new userids

```
USER CFCZOSAO  password  256M 512M G
CPU 00 BASE
CPU 01
OPTION CFVM
SHARE RELATIVE 1000
XAUTOLOG CONZOSAO
CONSOLE 009 3270 T concfvm

USER CONZOSAO  password  8M 32M G
   INCLUDE SECCON
```

Note that the `concfvm` value in CFCZOSAO DIRECT is not updated as a part of the DIRM ADD command, so the system programmer would need to do the following to correct the directory entry:

1. Run the command **`DIRM FOR CFCZOSAO GET`**.
2. Receive the resulting file to a minidisk in the CMS search order.
3. Edit the file using the command **`XEDIT CFCZOSAO DIRECT`**.
4. Change every occurrence of concfvm to CONZOSAO using the command:

   `CHANGE /concfvm/CONZOSAO/ * *`
5. Save the file using the command **`FILE`**.

6. Place an updated version of the directory entry into the source directory:

```
DIRM FOR CFCZOSAO REPLACE
```

The MVSDASD 0191 minidisk is a CMS-formatted minidisk containing EXECs used by each z/OS virtual machine to enable it to IPL as desired. Since this disk is shared by all the z/OS virtual machines, these EXECs should be designed to be generic, or should contain selection logic to control which statements are executed by various virtual machines. A sample PROFILE EXEC that could be used to choose a release to IPL from among a list of available releases is contained in Appendix A, “Sample PROFILE EXEC” on page 19.

z/OS

The basic z/OS environment described in this section includes the following:

- Each z/OS virtual machine includes access to a set of DASD volumes collectively referred to as the system residence volumes.
- Each z/OS release uses three 3390-3 volumes or one 3390-9 volume for its system residence volumes.
- Each z/OS virtual machine requires access to sufficient DASD to contain paging and spooling space for that virtual machine, plus space to write data that is usually considered private to the system (/etc and /tmp especially).
- Program products can be installed on the system residence volumes, or on separate volumes; they could be installed in existing z/OS SMP/E datasets or in one or more separate SMP/E datasets. Installing on the z/OS system residence volumes or in the z/OS SMP/E datasets would save on disk space, but would require these program products to be re-installed for each new release. If sufficient DASD is available, install program products on separate volumes and in separate SMP/E datasets to avoid having to re-install.
- All z/OS virtual machines share a common virtual device configuration, such that a single IODF file can be maintained and used by all the z/OS virtual machines. In addition, the dataset containing the IOCP definitions should be maintained on a separate volume from the system residence volumes, such that multiple versions of IOCP definitions do not have to be maintained. The device address of this volume then becomes the first part of the value specified in the LOADPARM parameter of the CP IPL command.
- If ease of administration and maintenance are important, all z/OS virtual machines should be similar in structure and appearance. That is, they should use similar dataset naming conventions, PARMLIB/PROCLIB structures, data storage conventions, and so on.
- When running as a guest virtual machine on z/VM, z/OS images have a default system name that is identical to the z/VM userid. In our running example, the system name would be ZOSAO.

There are several design points to consider when setting up such an environment, mostly centered around how much DASD is available for this environment and what level of DASD sharing among virtual machines the installation feels comfortable implementing. The system residence volume for each release of z/OS requires either three 3390 Model 3 volumes or one 3390 Model 9; in addition, SMP/E will require two 3390 Model 3s. Each instance of a z/OS virtual machine will require dedicated DASD for paging, spooling, and private data; one or two volumes of 3390 Model 3 would suffice in an environment that has little need for private storage. Therefore, in an environment where no sharing is desired, each image would need the equivalent of at least seven 3390 Model 3 volumes; in an environment that wants to share the SYSRES, each image needs two volumes of 3390 Model 3, plus the equivalent of

five volumes of 3390 Model 3 that are shared. Some additional items to consider are described in the following sections.

If the environment includes a limited amount of DASD

If the installation has very little DASD, then configuring the environment to share as many volumes as possible is very beneficial.

System residence volume

A single copy of the z/OS system residence volume can be shared among multiple test images to reduce the need for DASD space on each image.

- A single master copy of the system residence is set up and maintained by one particular z/OS virtual machine, referred to in the following discussion as the ZOSMAINT machine.
- Each test virtual machine has read-only links to this DASD. This implies that each virtual machine sees exactly the same level of code as other virtual machines, provided that no STEPLIBs or other methods of overriding system code are used. Further, when updates are applied to the system residence volume, all users will see those changes at next IPL.

PARMLIB structure

In a shared-DASD environment, each image needs to have configuration information that is specific to that instance of z/OS (for example, TCP/IP profiles so that each image gets a unique IP address). Shared PARMLIBs can be used to contain the bulk of the configuration information, with support team members being the only users permitted to alter this configuration information, and only on the ZOSMAINT machine. However, each user tends to have to make minor changes to his or her z/OS instance, so a PARMLIB hierarchy can be implemented to establish a private PARMLIB for each image as the first one to be checked. These private PARMLIBs can reside on DASD that is private to the particular image, or on DASD that is shared R/W by all images. PARMLIB search order is specified in the LOADxx member used at IPL time. Example 10 shows a typical LOADxx member (sample file SYS1.PARMLIB.ZOSR8.LOADAO in the additional material described in Appendix B on page 23).

Example 10 Sample LOADxx member

```
IEASYM   (08,AO)                 <-- this image uses IEASYMAO
IODF     92 SYS1
NUCLEUS  1
SYSCAT   SYS00214  SYS1.MCAT390
PARMLIB  ZOSAO.PARMLIB           <-- image-specific searched 1st
PARMLIB  SYS1.PARMLIB.ZOSR8      <-- release-specific
PARMLIB  SYS1.PARMLIB.ZOS        <-- applies across all releases
PARMLIB  SYS1.PARMLIB            <-- contains only LOADxx members
SYSPARM  (AO,08,L)
```

This LOADxx member needs to be in dataset SYS1.PARMLIB located on the same volume that contains the IODF dataset (SYS1.IODF92 in this example). The IEASYM statement identifies that this IPL should use IEASYMAO to load image-specific variable definitions; if this member does not contain release-specific definitions, it could be contained in the PARMLIB that applies to all releases (SYS1,PARMLIB.ZOS in this example). Example 11 shows what such a member might look like (sample file SYS1.PARMLIB.ZOS.IEASYMAO in the additional material described in Appendix B on page 23).

Example 11 Sample LOADxx member

```
SYSDEF  SYSCLONE(A0)
```

Also in Example 10, the SYSPARM statement identifies that this IPL should use IEASYSAO and then IEASYS08 to load image-specific system parameters. An example of how such a member might look is Example 12 (sample file SYS1.PARMLIB.ZOSR8.IEASYSAO.NOPLEX in the additional material described in Appendix B on page 23).

Example 12 Member that contains image-specific system parameters

```
CMD=(00,A0),
LNK=(A0,00,L),
LPA=(00,L),
PROD=00,
PROG=(00,L),
SMS=95,
OMVS=(00,A0,PD,LZ)
```

With the PARMLIB search order specified, default members go into SYS1.PARMLIB.ZOS, members that have different values for each release go into release-specific PDSes like SYS1.PARMLIB.ZOSR8, and members that need to be tailored for each z/OS instance go in the instance-specific PDS (ZOSAO.PARMLIB in this example).

An example of a set of configuration parameters that generally does not need to be changed from release to release or from instance to instance is the CLOCKxx member, which defines how system clocks are initialized. An example of CLOCK00 is shown in Example 13 (sample file SYS1.PARMLIB.ZOS.CLOCK00 in the additional material described in Appendix B on page 23).

Example 13 Example of the CLOCKxx member

```
OPERATOR NOPROMPT
TIMEZONE W.04.00.00
ETRMODE  NO
ETRZONE  NO
ETRDELTA 10
SIMETRID 00
```

Most images will not need to change clock values; however, if one does need to, in order to test date- or time-sensitive programs or systems or to operate in a different time zone, there are a few ways to enable this:

- Create a new CLOCKxx member in the image-specific PARMLIB, with value OPERATOR PROMPT; add `CLK=xx` to the image's IEASYS parmlib member; and respond to operator prompts to set date and time at IPL of the virtual machine.
- Use existing defaults to IPL the virtual machine, then issue the command:

  ```
  SET DATE=yyyy.ddd,TIME=hh:mm,TIMEZONE=z.hh.mm
  ```

  This z/OS command is described in detail in IBM publication *z/OS MVS System Commands,* SA22-7627.
- Before the virtual machine is IPLed, modify the virtual clock by issuing the CP command SET VTOD DATE mm/dd/yyyy TIME hh:mm:ss. Details of this command are provided in IBM publication *z/VM CP Command and Utility Reference*, SC24-6008.

Note: In order to override the z/VM system time in the virtual machine, OPTION TODENABLE is required in the virtual machine's directory entry. Also, if your z/VM real processor does not have access to a SYSPLEX TIMER or other external time source, then you may find that you need to modify CLOCKxx parmlib entries (and z/VM's SYSTEM CONFIG file) for each Daylight Savings Time adjustment, if your location observes such changes.

An example of a configuration parameter that generally needs to differ from release to release but does not need to be changed from instance to instance is the UNIX® System Services definition member, BPXPRMxx. In particular, the /usr directory tree, which contains the basic set of programs supplied by IBM for the UNIX System Services environment, necessarily differs from release to release, so the mount of this /usr filesystem can differ from one release to another. Example 14 shows a statement that might be found in a release-specific BPXPRM08 member.

Example 14 Statement that may be found in BPXPRM08

```
MOUNT  FILESYSTEM('OMVS.MNT.ZOSR8.HFS')
         TYPE(HFS)
         MODE(READ)
         MOUNTPOINT('/usr')
```

This entry, which would exist in release-specific PDS SYS1.PARMLIB.ZOSR8 in our example, mounts the release-specific /usr filesystem read-only at /usr.

An example of a configuration parameter that generally differs from instance to instance but may be identical from release to release for each image is the set of parameters in IEASYSxx, which are used to define a parallel sysplex. This setup normally is identical from one release to the next, but the definitions may need to be different from one sysplex to another. An example of such IEASYSxx statements is shown in Example 15 (sample file SYS1.PARMLIB.ZOSR8.IEASYSAO.WITHPLEX in the additional material described in Appendix B on page 23).

Example 15 Sample IEASYxx statements

```
CMD=(00,A0),
CON=75,
COUPLE=75,
GRS=STAR,
GRSCNF=00,
GRSRNL=01,
LNK=(A0,00,L),
LPA=(00,L),
PLEXCFG=MULTISYSTEM,
OMVS=(00,A0)
```

This example indicates the following configuration files would be included:

- COMMNDAO is an image-specific member that includes commands to be run on the instance at startup; this member would probably reside in ZOSAO.PARMLIB if it does not contain release-specific values, or in SYS1.PARMLIB.ZOSR8 if, for example, it contains commands specific to z/OS Release 8.
- BPXPRM00 is the initial UNIX System Services configuration member to be used, and probably includes MOUNT statements for the release-specific /usr filesystem, so it would be contained in PDS SYS1.PARMLIB.ZOSR8.

- BPXPRMAO would then be used to further configure the UNIX System Services environment by, for example, mounting filesystems specific to the z/OS instance; these filesystems would probably not have to differ from release to release. For example, /etc holds configuration information for TCP/IP and does not need to change from release to release, so it would probably reside in the image-specific PDS ZOSAO.PARMLIB.

TCP/IP configuration

One final set of information needed to distinguish one instance from another is the TCP/IP profile. In general, z/OS running in a z/VM virtual machine will get its &SYSNAME system variable from the userid of the virtual machine in which it was IPLed. For example, if the z/VM userid is ZOSAO then z/OS will default its &SYSNAME variable to ZOSAO. This system variable can then be used to locate PARMLIB information specific to that z/OS instance. If, for example, you want to use a single TCP/IP startup PROC, pointing to a single dataset in which you maintain all the TCP/IP profiles for each z/OS instance, your startup PROC might contain a PROFILE statement similar to:

```
//PROFILE  DD DISP=SHR,DSN=TCPIP.PROFILE(&SYSNAME)
```

In this statement &SYSNAME is used to find the profile for this instance in the partitioned dataset. Continuing the previous examples, if the z/VM userid is ZOSAO, then z/OS looks for member TCPIP.PROFILE(ZOSAO) to find the TCP/IP profile for this instance. This profile contains all the statements necessary to start a TCP/IP stack on the instance. Example 16 shows a sample TCP/IP profile (sample file TCPIP.PROFILE.ZOSAO in the additional material described in Appendix B on page 23).

Example 16 Sample TCP/IP profile

```
ARPAGE 2

UDPCONFIG UDPSENDBFRSIZE 65535 UDPRCVBUFRSIZE 65535 NOUDPQUEUELIMIT

DATASETPREFIX SHR

AUTOLOG
    TN3270      ; TELNET server
    CSFTPD      ; FTP Server (new name, old was FTPSERVE)
    SYSLOGD     ; SYSLOG daemon
    USINETD     ; INET daemon
    CSFTPOE     ; FTP OE Server
;   MVSNFS      ; Network File System Server
ENDAUTOLOG

PORT
    20 TCP OMVS      NOAUTOLOG ; OE FTP Server
           DELAYACKS
    21 TCP OMVS                ; OE FTP Server
;   22 TCP OMVS                ; OE SSH Server
    23 TCP OMVS                ; OE Telnet Server
   111 TCP OMVS                ; OE Portmap Server
   111 UDP OMVS                ; OE Portmap Server
   161 UDP SNMPD               ; SNMP Agent
   162 UDP SNMPQE              ; OE SNMPQE Agent
   446 TCP OMVS                ; DRDA SQL port for DB2C
   512 TCP OMVS                ; OMVS rexec
   513 TCP OMVS                ; OMVS rlogin
   514 TCP OMVS                ; OMVS rshd
```

```
;  515 TCP LPSERVE            ; LPD Server
   520 UDP ROUTED             ; RouteD Server
   534 TCP OMVS               ; OMVS FTP
   535 TCP OMVS               ; OMVS FTP
   580 UDP NCPROUT            ; NCPROUTE Server
   623 TCP OMVS               ; OMVS TELNET SERVER
   750 TCP MVSKERB            ; Kerberos
   750 UDP MVSKERB            ; Kerberos
   751 TCP ADM@SRV            ; Kerberos Admin Server
   751 UDP ADM@SRV            ; Kerberos Admin Server
  2049 UDP MVSNFS             ; NFS Server
    3000 TCP CICSTCP            ; CICS Socket
  5020 TCP OMVS               ; DRDA resync port for DB2C

DEVICE OSAFE  MPCIPA   NONROUTER AUTORESTART
LINK   OSAFE  IPAQENET OSAFE VLANID 541

INTERFACE QD1 DEFINE IPAQENET6 PORTNAME OSAFE NONROUTER
              IPADDR 2002:90C:1401:541:9:12:47:25

HOME
   9.12.47.25     OSAFE

BEGINROUTES
     ROUTE  9.12.47.0/24                    = OSAFE   MTU 1500
     ROUTE  2002:90C:1401:541:9:12:47:0/120 = QD1     MTU 1500
     ROUTE  DEFAULT             9.12.47.1   OSAFE   MTU 1500
ENDROUTES

IPCONFIG NODATAGRAMFWD

START OSAFE
START QD1
```

Generally the only change needed to this profile is in the HOME statement for IPv4 links or in the INTERFACE statement for IPv6 links.

If the environment includes a sufficiently large amount of DASD

If the installation has lots of DASD, then configuring the environment to not share DASD volumes may be more expedient. A single master copy of the system residence is set up (initially) and maintained using the ZOSMAINT machine.

Each new image is given a copy of the system residence:

- If the DASD supports hardware FLASHCOPY, then these copies can be created very quickly, on the order of a minute or so for a 3390-9. This can be used, for example, every time the z/OS test machine IPLs to ensure that it is running on the absolute latest version of the z/OS system residence volumes.
- If the DASD does not support hardware FLASHCOPY, some other mechanism of duplicating DASD volumes (for example, z/VM's DDR utility) must be used and might take longer to create copies. This might imply that individual copies of the system residence volumes will be updated less frequently.

Each new instance can use identical PARMLIB members from one release to another (with the obvious exception of changes needed for TCP/IP home addresses) because each instance has its own copy of DASD. Thus, once the configuration is set up initially, very few changes are required to create a new instance.

The installation might choose to define a z/OS image with all necessary applications installed, and an initial set of data used to test a particular application being developed, then use the FLASHCOPY approach to quickly build a new instance of the z/OS image; the tester would interact with the instance for as long as needed, then shut the instance down and log it off, at which point it is effectively destroyed. The next time the tester needs an instance, the same steps would be used to recreate the instance. If there is data that needs to be kept available from one instantiation to the next, then a separate set of disks could be maintained and connected to the virtual machine as needed.

Creating an environment

To construct a z/OS guest environment on z/VM, proceed as follows:

1. Analyze z/OS application and middleware testing needs to determine the set of z/OS images to create and what customization is needed at the levels suggested in this document.
   a. Determine the amount of DASD available and the best provisioning approach to use.
   b. Identify the parts (Application hosting runtimes, IBM program products, third party products) needed in the z/OS images.
   c. Decide if multiple releases of products will be required, either initially or in the future.
2. Create the VSWITCH definition for network connectivity, if needed, and add to the z/VM SYSTEM CONFIG file.
3. Create or edit the prototype directory entries for the virtual machines that will host z/OS images and add them to the z/VM User Directory.
4. Create z/OS images set up for additional customization to act as masters for the provisioning process. Include the following, considering whether to install them on the SYSRES volume or on separate disks:
   a. SYSRES with PARMLIB members
   b. Paging space
   c. JES and SPOOL datasets
   d. TCP/IP profiles
   e. Other IBM products, ISV products
   f. Test databases or files
5. Create virtual machine definitions for the test environments.
6. Create z/VM profile EXECs to access the shared minidisks and install them either on the z/OS test machine A (191) disks or on a shared minidisk available to all guests.
7. Create EXECs to copy the master volumes to create each flavor of virtual machine. Use FLASHCOPY or DDR to do this.
8. Create user-level PARMLIBs for user customization using a z/OS maintenance virtual machine.
9. Test the creation EXECs.

10. Develop a pick list (a REXX EXEC) to allow a user to define the test environment he wants based on selecting components.
11. Form the names of the EXECs that instantiate z/OS images based on a user's selection.
12. Create EXECs to IPL the selected images.

Using the environment

To use a z/OS guest environment on z/VM for the first time, proceed as follows:

1. Log on to z/VM.
2. Run the REXX EXEC created in step 10 in the previous procedure (Creating an environment) to display the list of selection criteria.
3. Select the configuration to test by picking appropriate selection criteria. (Assume this runs an EXEC to create it.)
4. Manually IPL the virtual machine if the selection criteria EXEC does not automatically perform the IPL.
5. Perform any required administrative functions on the test environment to make it usable.
6. If required, build and install applications in the test environment.
7. Conduct the tests.
8. Perform post-test activities (for example, print dumps).
9. Close down the system and its subsystems.
10. Log off z/VM.
11. If the virtual machine will continue to be used, no additional commands are needed to preserve it. If the virtual machine is no longer needed, the system programmer will need to remove the userid from the VM DIRECTORY using the DIRM PURGE command.

To resume using a previously created virtual machine, proceed as follows:

1. Log on to z/VM.
2. Manually IPL the virtual machine if the virtual machine directory entry or PROFILE EXEC used by this virtual machine does not automatically perform the IPL.
3. If required, build and install applications in the test environment.
4. Conduct the tests.
5. Perform post-test activities.
6. Close down the system and its subsystems.
7. Log off z/VM.
8. If the virtual machine will continue to be used, no additional commands are needed to preserve it. If the virtual machine is no longer needed, the system programmer will need to remove the userid from the VM DIRECTORY using the DIRM PURGE command.

Implementation notes from the z/OS middleware development team at IBM Poughkeepsie, NY

Our multiple VM systems use a shared z/VM directory and DIRMAINT (with satellite DIRMSAT service virtual machines on each VM system) to control the various virtual machines. We have chosen not to implement RACF® at this time, to make maintenance and support easier.

The central design behind each z/OS test image is the ability to cheaply and easily create a new image, and to be able to maintain the entire pool of z/OS test images with as small a system programming staff as possible, using older-generation “hand-me-down” hardware. In order to accomplish this, we have designed a system image with mostly shared R/O DASD among all the various test images. Such volumes as SYSRES and program product disks are read-only so that no one can alter what others see. The reason we use shared packs instead of separate copies for each test image is to avoid maintaining all those copies while minimizing the amount of time needed to set up an image. Each image also receives a couple of private R/W DASD volumes to use for private data, spooling, paging, and dump space.

Note: One approach to consider is to have the virtual machine “cloned” to a completely new set of R/W DASD each time you want to IPL it. That way you get the latest version of the system, and can modify it as needed for your particular test environment without affecting others. The drawback is that it takes time to copy each of the volumes necessary. FLASHCOPY control units will significantly speed this up, making this approach more workable. Since our current DASD does not support FLASHCOPY, we chose not to make clones at each guest IPL.

In order to maintain separate configuration data for each of the test images, each image has a name that begins with the same characters (“ZOS” in the previous examples) but is assigned a unique 2-character suffix, and each image gets its own PARMLIB and PROCLIB. LOADxx members are set up to specify the image’s private PARMLIB as the first dataset to search when looking for IPL parameters, followed by a PARMLIB that is shared by all the test images.

Note: We add a wrinkle to our environment by supporting multiple releases, as well as SYSPLEXes, so we have a PARMLIB for each release, where we put all the configuration information for each test image authorized for that release. Each sysplex also gets a PARMLIB to contain parameters that are common to all its members. Thus, our PARMLIB search order ends up being:

- image-specific PARMLIB
- sysplex-wide PARMLIB
- release-specific PARMLIB
- generic z/OS PARMLIB

PROCLIBs work similarly to PARMLIBs, except that we do not maintain separate release-specific PROCLIBs at this time.

In order to maintain all these PARMLIBs and PROCLIBs, we put the private ones on shared packs that are backed up regularly. We also have a z/OS image that is designated as our “maintenance” image, where we do all the initial configuration changes and day-to-day service on the various test images. Each image has READ/WRITE access only to the PARMLIBs specific to itself or its sysplex; if users need to override parameters normally contained in the higher-level PARMLIB structures, they can copy the desired PARMLIB

members to the plex- or image-specific PARMLIB and change as desired. Furthermore, we have chosen to implement the image-specific and plex-specific PARMLIBs on DASD that is shared R/W by all images, rather than on DASD that is private to the image or sysplex; this simplifies recovery from problems due to a PARMLIB change that causes the image or plex to no longer be IPLable. An added benefit is that users can make their own local configuration changes without getting the system programming staff involved. This works well, and our people are disciplined enough not to alter information for images they do not own – plus, they are able to learn from configuration changes made by other team members.

TCP/IP connections are provided through use of a VSWITCH that has access to the corporate network through a QDIO OSA Express adapter. IP addresses are assigned by our networking team, and are maintained in a dataset to which each test image has READ access for holding the TCP/IP profiles. Again, this dataset is maintained on our maintenance image.

We also have a need to create data or programs on one image to be shared with others, in an ad-hoc fashion. To support this, we created a pool of DASD we refer to as HIGHRISK, along with a set of rules for high-level dataset qualifiers, to allow materials created in this pool to facilitate the needed sharing. One other use for this DASD pool is to store dumps created on any test image so that they can be debugged on another image without the need to ship the dumps around. The same argument can be made for sharing data and test cases in this fashion.

z/VM does provide the capability to create a sysplex using simulated coupling facility hardware, and it is quite easy to do. You can create DASD-only loggers, single-image SYSPLEXes, and multiple-image SYSPLEXes with up to 32 images in the plex. You can define your plex to have a single coupling facility or multiple ones.

With the shared DASD and PARMLIB structures we have, we can also add program products (such as DB2®) easily to specific images as needed.

The team that wrote this IBM Redpaper

This paper was produced by a team of specialists working at the International Technical Support Organization, Poughkeepsie Center.

Alan Kline is a Software Engineer in the IBM System z® Operating System Development organization in Poughkeepsie, New York. He has worked at IBM for 29 years. He holds a Master's Degree in Computer Science from Syracuse University and Bachelor of Science in Mathematics from Gannon University. His areas of expertise include system programming support for z/VM, z/OS and z/Linux.

John Kapernick is an IBM Senior Technical Staff Member in the United States. He has worked for IBM for 42 years in assignments in the field, headquarters, and development. He works in z/OS Strategy and Design. He has degrees in Industrial Engineering from Georgia Tech and Distributed Systems Engineering from Polytechnic University. He works primarily with the products of the application development and runtime stack of z/OS.

Romney White is an IBM Senior Technical Staff Member in the United States. He has over 40 years of experience in software design and development and holds degrees in Mathematics and Computer Science from the University of Waterloo, Canada. His areas of expertise include mainframe virtualization and he has written and presented extensively on a variety of topics in this field.

Reed Mullen is a Consulting Programmer in the IBM System z organization in Endicott, New York. He has worked at IBM for 27 years. His areas of expertise include mainframe virtualization and server consolidation. He has written extensively on System z virtualization technology.

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

Lydia Parziale
International Technical Support Organization, Poughkeepsie Center

Martha McConaghy
Marist College, Poughkeepsie, NY

Mike Walter
Hewitt Associates

Jim Vincent, Rick Barlow
Nationwide Insurance

Dave Crow, Val Christensen, Ed Webb, and Dan Squillace
SAS Institute

A

Sample PROFILE EXEC

This sample PROFILE EXEC contains a small section designed to connect a vCTCA between the z/OS instance and your RSCS networking virtual machine. If you choose to implement this connectivity, you will need to provide updates to your RSCS configuration as well. These files, located on RSCS 191 minidisk, include PROFILE RSCS (which contains statements to define devices for connection to z/OS instances) and RSCS CONFIG (which configures the connections). This paper does not describe an RSCS implementation; for more information about setting up an RSCS network among real and virtual machines, refer to IBM publication *Virtual Machine Remote Spooling Communications Subsystem Networking Operation and Use*, SH24-5220.

See Appendix B, “Additional material” on page 23 for instructions on how to download a copy of this file.

Example 17 Sample PROFILE EXEC on MVSDASD.191 minidisk

```
/*************************************************************/
/*  PROFILE EXEC    ON MVS191 191 DISK                       */
/*  MASTER PROFILE EXEC FOR ALL z/OS MACHINES                */
/*  CREATED   04/27/95                                       */
/*************************************************************/
If userid() = 'MVSDASD' then
  do
    'CP SPOOL CONS START'
    say 'Hello, MVSDASD !!'
    /* set some PF keys */
    'CP SET PF2  IMMED RDRLIST'
    'CP SET PF5  IMMED FILELIST * * A'
    'CP SET PF12 RETRIEVE'
    'CP SPOOL CONS STOP CLOSE'
    exit
  end
```

```
SAY '*************************************************************'
SAY '*    Welcome to z/OS Image ' userid()
SAY '*************************************************************'
'CP SET RUN ON'
'CP SET PF12 RETRIEVE BACKWARD'   /* set pf key to retrieve  */
'CP PURGE RDR ALL'

/*************************************************************/
/* DEFINE GRAFS TO ENABLE SIMULATEOUS LOGONS                 */
/*************************************************************/
'DEFINE GRAF 684'
'DEFINE GRAF 685'
'DEFINE GRAF 686'
'DEFINE GRAF 687'
'DEFINE GRAF 688'
'DEFINE GRAF 689'
'DEFINE GRAF 68A'
'DEFINE GRAF 68B'
'DEFINE GRAF 68C'
'DEFINE GRAF 68D'
'DEFINE GRAF 68E'
'DEFINE GRAF 68F'

/************ Virtual CTC section ****************************/
/* this section would contain vCTCA defs for such things as  */
/* RSCS/NJE links                                            */
SELECT
  WHEN USERID()=DCEDDE2
    THEN DO
       'CP COUPLE 0500 RSCS     0505'
     END
  WHEN USERID()=ZOSAO
    THEN DO
       'CP COUPLE 0500 RSCS     0506'
     END
  OTHERWISE NOP
END

/***************** LOADPARM section ***************************/
CFVM = ''   /* clear CF virtual machine name for none defined */
SELECT
  WHEN USERID()=DCEDDE2  THEN PARM='AMM1'
  WHEN USERID()=ZOSAO  THEN PARM='AOM1'
  OTHERWISE PARM=''
END
/************ Ending LOADPARM section *************************/

/***************** SYSPLEX section ****************************/
CFVM = ''   /* clear CF virtual machine name for none defined */
SELECT
  WHEN USERID()=ZOSAO  THEN CFVM=CFCDDE4
  OTHERWISE NOP
END
```

```
IF CFVM <> '' THEN
  DO
    select
      when userid()=ZOSAO then 'CP DEF MSGP CFC1DDE4 VDEV FA4'
      otherwise
        NOP
      end
  END
/************ Ending SYSPLEX section *************************/

/************ IPL section ************************************/
/*  empty the input queue  */
do i = q to queued()
  pull garbage
end

try_rel_again:
/*  Find out what release the user wants to IPL  */
say 'Valid releases include:'
say '   Z8  = z/OS R8'
say '   Z9  = z/OS R9'
resp = ''
parse upper pull resp garbage
select
  when resp = 'Z8' then
    do
      'CP DET 0400-0402'  /* we will use 400-402 as our SYSRES address */
      'CP LINK MVSDASD 801 400 RR'
      'CP LINK MVSDASD 892 401 RR'
      'CP LINK MVSDASD 893 402 RR'
    end
  when resp = 'Z9' then
    do
      'CP DET 0400-0402'  /* will use 400-402 as SYSRES addr */
      'CP LINK MVSDASD 901 400 RR' /* z/OS R9 is single 3390-9 */
    end
  otherwise
    do
      say 'You entered an invalid value! Enter "Z8" or "Z9"!'
      signal try_rel_again
    end
end /* SELECT */

/* set some PF keys for use second-level */
'CP TERM BRKKEY PF03'

/* now do the IPL */
'CP SET SVC76 VM'
NL='15'X
say 'IPL command = IPL 0400 CLEAR LOADPARM 04F5'PARM
'CP TERM CON 3270'||NL||'IPL 0400 CLEAR LOADPARM 04F5'||PARM

exit
```

B

Additional material

This paper refers to additional material that can be downloaded from the Internet as described here.

Locating the Web material

The Web material associated with this paper is available in softcopy on the Internet from the IBM Redbooks® Web server. Point your Web browser to:

ftp://www.redbooks.ibm.com/redbooks/REDP4507.zip

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the IBMRedpaper form number, REDP4507.

Using the Web material

The additional Web material that accompanies this paper includes the following files:

File name	*Description*
REDP4507.zip	Zipped samples that follow the examples in the book

System requirements for downloading the Web material

The following system configuration is recommended:

Hard disk space:	2 MB minimum
Operating System:	At least Windows® XP
Processor:	1700 MHz or higher
Memory:	At least 500 MB

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this folder.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

This document REDP-4507-00 was created or updated on April 10, 2009.

Send us your comments in one of the following ways:
- Use the online **Contact us** review Redbooks form found at:
  ibm.com/redbooks
- Send your comments in an email to:
  redbooks@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400 U.S.A.

Trademarks