Customizing a migrated z/OS

Contents

Customizing a migrated z/OS system

Learn how to customize a migrated z/OS® system.

A brief introduction to z/OS system configuration

In the simplest case, z/OS is configured by changing partitioned data set (PDS) members in the data sets SYS1.PROCLIB, SYS1.PARMLIB, and a few other important data sets including site-specific partitioned data sets.

Most configuration (parmlib) member names consist of a predefined name with a two character suffix added. A common convention is to refer to the suffix as *xx*, so you often see references to LOADxx, IEASYSxx, and so on. Configuration files refer to other members by a keyword and suffix number. For example, a member that is called IEASYSRC might identify the main z/OS UNIX configuration file with the line OMVS=RC, which means that z/OS UNIX finds the member that contains its configuration parameters by starting with a predetermined name, BPXPRM, and add the suffix RC (resulting in member name of BPXPRMRC). The keywords in the configuration files *are not* usually the same as the member name prefixes.

Data sets such as SYS1.PARMLIB and SYS1.PROCLIB are not updated directly. Some installation-specific libraries are searched before the SYS1 data sets, leaving the SYS1 libraries with IBM® supplied defaults.

The z/OS ADCD for z Systems™ Development and Test Environment defines several sets of alternative libraries. The configurations for the distribution itself are stored in a set of libraries that start with the qualifiers ADCD.*. This z/OS distribution also provides a set of libraries for you to use, which start with the high-level qualifier USER. This z/OS distribution is already set up to read from most of the USER libraries before the ADCD.* libraries. For more information on the configuration of the z/OS ADCD for z Systems Development and Test Environment, see the specific sessions Customizing the May 2017 Edition z/OS 2.2 ADCD for z Systems Development and Test Environment.

If you are migrating your own z/OS distribution, you probably defined your own alternative libraries to contain any z/OS customizations. Your system administrator knows the library structure that is defined for your z/OS distribution, and what libraries to use to contain any customizations. For purposes of describing customizations, this IBM Knowledge Center uses USER.* libraries, which tend to be commonly available in any z/OS distribution. When you follow these instructions, make all changes in USER.* libraries, except for adding LOADPARM members to SYS1.IPLPARM.

Note: Subsystem configuration changes, such as CICS® definitions, also change original libraries such as the CICS CSD.

The IPL process: LOADxx and IEASYSxx members

When z/OS is started, it looks in data set SYS1.IPLPARM for a member called LOADxx. The xx value is specified in the IPL statement of the start script that was created. A LOADxx member defines various settings to start the system, such as the parmlib concatenation that indicates which data sets are to be searched for other configuration members. The LOADxx member also defines which IEASYSxx member

is to be used. IEASYSxx is considered the starting point for system configuration because it contains pointers to other parmlib members that are used during the IPL process.

Tip: Create a loadparm that starts the system without pointing to any alternative libraries that contain customized parmlibs, proclibs, and so on. By creating a loadparm if you make a mistake in your customizations that causes z/OS to not start, you can try to perform an IPL of the system with the loadparm that does not point to the alternative libraries. This method gives you TSO access, where you can modify the customizations to correct any mistakes.

PROCLIB: System procedure library

Parmlib members contain only configuration information. The procedures that start the various subsystems and servers are found in a different concatenation called PROCLIB. As an example, a z/OS ADCD for z Systems Development and Test Environment contains SYS1.PROCLIB, ADCD.Zxxx.PROCLIB, and USER.PROCLIB in its PROCLIB concatenation. To include any PROCLIB data set in the PROCLIB concatenation, you must modify the MSTJCLxx member of parmlib and your JES procedure.

For more information, see "Enabling use of USER.PROCLIB and IPLing" on page 11.

z Systems serial numbers

z Systems CECs have unique serial numbers, allowing software to identify the machine and LPAR.

The z Systems serial number is part of its CPU ID, a concatenation of the serial number, LPAR number, and other information. In the simple environment that is described in this IBM Knowledge Center, which is a single USB hardware device that is connected to a single PC hosting z Systems Development and Test Environment, the serial number of the z Systems created by zPDT® is the serial number of the USB hardware device. When a product license server or a license manager is used, the serial numbers are randomly generated by the UIM server that is running as part of the product license server, and assigned to the z Systems Development and Test Environment instance.

Tip: From the System console or TSO SDSF, a Display M command returns all assigned machine serial numbers. This information is also available by using the CSRSI callable service, which is documented in *MVS™ Programming: Callable Services for High-Level Languages Version 2 Release 1*, SA23-1377.

After it is assigned, whether from the local USB Hardware Device or from a license server, the z Systems serial number remains consistent, even across upgrades to new emulator versions, and when new license files are applied to the USB hardware device. This assignment allows for a consistent CPU ID. That first assigned serial number is stored locally on any z Systems Development and Test Environment instance in /etc/z1090/uim/uimclient.db and reused, thus allowing for this consistency even when you are switching between using a local USB Hardware Device and a license server.

The serial number can change in a few cases. The most common case is if you enter a uimreset -l command, which causes the serial number to be reassigned from either the local USB hardware device or from a license server. It can also change if you somehow delete or corrupt the uimclient.db file. For more information about z Systems CEC serial numbers and zPDT, see paragraph 8.1 "Methodology" in the zPDT Guide and Reference.

Tip: If consistent serial numbers or CPU IDs are a requirement, back up your `/etc/z1090/uim/uimclient.db` file after your initial z Systems Development and Test Environment activation. If the file is corrupted or changed by a **`uimreset -l`** command, it can be restored from the backup and thus restore the original serial number.

Configuration overview

Learn how to configure the z/OS system so that you isolate the customized data in your z/OS distribution volumes and establish TCP/IP communications.

These configuration instructions explain how to:

- Create new z/OS UNIX file systems for `/tmp` and `/u`.
- Customize TCP/IP settings to establish network connectivity.

You might also want to make some additional changes that are commonly made.

- Create TSO user IDs.
- Customize ISPF defaults and the ISPF main panel
- Change console defaults
- Streamline startup and shutdown scripts
- Create an NJE connection to existing z/OS systems

Place these customizations on a single volume, such as ensuring your USER.* libraries are on one volume. Placing these customizations on a single volume helps with future migrations to new versions of z/OS distributions.

Logging on to TSO

After you IPL, you can use the x3270 emulator on the host Linux to start a TSO session

with the following command:

```
x3270 -port 3270 localhost &
```

An alternative format of the x3270 command, which produces a larger screen size, is

```
x3270  -model 4 localhost:3270 &
```

You can also use an emulator that is not native to the host Linux. For more information, see “Logging on to TSO from emulators not native to the host Linux” on page 12.

Log on to TSO with a valid user ID in your z/OS distribution. You might want to make a few minor changes to your ISPF session before you start working. If you are accustomed to using TSO naming conventions in ISPF, then to ensure that you do not write out files with unexpected high-level qualifiers, enter this command:

```
TSO PROFILE PREFIX(tso user id)
```

Creating new zFS files for `/tmp` and `/u`

Management of z/OS UNIX file systems in z/OS is a complex area. This procedure provides a simplistic design for new file systems. Considerations such as space requirements and alternative mount points might require a more in-depth plan.

Your z/OS distribution might provide fairly small file systems for the /tmp and /u directories. Small file systems can cause problems, particularly when you are installing z/OS software by using SMP/E, or when programs create large memory dumps on the z/OS UNIX file system. Allocate 100 primary cylinders and 20 secondary cylinders for these file systems. If the current file systems for /tmp and /u are not large enough, replace them. To create new file systems for /tmp and /u, create two new zFS files and associate these new files with the /tmp and /u directories. Catalog these two new zFS files on the same volume as your other customizations. By avoiding use of the master catalog during migration to a new z/OS distribution, you must only import the user catalog and replicate the alias definitions so that catalog entries are restored.

To create new zFS files for /tmp and /u:

- Create aliases for the high-level qualifiers of the zFS files
- Create the zFS files
- Modify the appropriate BPXPRMxx parmlib member to mount the new zFS files with the /tmp and /u directories
- Copy the existing /tmp and /u directories to the new /tmp and /u directories that are mounted with the new zFS files
- Optionally create a new load parm to contain these changes
- Shut down the system and re-IPL

Any z/OS ADCD for z Systems Development and Test Environment supplies samples with more specific instructions, including sample JCL. These samples can help you perform this step even if you are migrating your own z/OS distribution. For more information, see Customizing the May 2017 Edition z/OS 2.2 ADCD for z Systems Development and Test Environment.

In the examples in these topics, the two z/OS UNIX file systems are created with a high-level qualifier of CUST on sample volume S1CUST.

Creating ALIASes for the high-level qualifiers of the zFS files

Start by creating an alias for CUST and any other qualifiers that are used to create data sets here. Create the alias and the file systems on a volume with your other customizations so that it is easy to migrate to a new z/OS distribution.

When you create new users, also create an alias for the user ID in the same catalog. This sample JCL shows you how.

```
//Q12ALIAS JOB (ACCT),MSGCLASS=H,NOTIFY=&SYSUID.
//*-------------------------------------------------
//DEFALIAS EXEC PGM=IDCAMS,REGION=0M
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
    DEFINE ALIAS (NAME(CUST) RELATE(USERCAT.Zxxx.USER))
/*
```

Creating the zFS files

Use the JCL in this topic to create two new zFS files on volume xxCFG1 named CUST.ZFS.U and CUST.ZFS.TMP.

Adjust the space allocations as needed.

```
//Q13ZFS JOB 'IBMUSER',CLASS=A,NOTIFY=&SYSUID,MSGCLASS=H
//ZFSALLOC EXEC PGM=IDCAMS,REGION=0M
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
```

```
    DEFINE CLUSTER( -
             NAME(CUST.ZFS.TMP) -
             VOLUME(xxCFG1) -
             LINEAR -
             CYL(100 20) -
             SHAREOPTIONS(3) -
                  )
    DEFINE CLUSTER( -
             NAME(CUST.ZFS.U) -
             VOLUME(xxCFG1) -
             LINEAR -
             CYL(100 20) -
             SHAREOPTIONS(3) -
                  )
/*
//ZFSFORMT EXEC PGM=IOEAGFMT,REGION=0M,COND=(0,LT,ZFSALLOC),
//            PARM='-aggregate CUST.ZFS.TMP -compat'
//SYSPRINT DD SYSOUT=*
//ZFSFORMT EXEC PGM=IOEAGFMT,REGION=0M,COND=(0,LT,ZFSALLOC),
//            PARM='-aggregate CUST.ZFS.U -compat'
//SYSPRINT DD SYSOUT=*
```

Modifying BPXPRM00 parmlib member

After you create CUST.ZFS.TMP and CUST.ZFS.U, copy ADCD.Zxxx.PARMLIB(BPXPRM00) into USER.PARMLIB.

Comment out the existing mount of the zFS for tmp, and replace it with a mount of CUST.ZFS.TMP. For example,

```
/* MOUNT    FILESYSTEM('ZFS.&ADCDLVL..TMP') */
/*          TYPE(HFS)                        */
/*         MODE(RDWR) NOAUTOMOVE             */
/*         MOUNTPOINT('/&SYSNAME./tmp')      */

MOUNT    FILESYSTEM('CUST.ZFS.TMP')
         TYPE(ZFS)
         MODE(RDWR) NOAUTOMOVE
         MOUNTPOINT('/&SYSNAME./tmp')
```

Make a similar change for the mount of /u:

```
/* MOUNT    FILESYSTEM('ZFS.&ADCDLVL..USERS') */
/*         TYPE(HFS)                 */
/*         MODE(RDWR) NOAUTOMOVE     */
/*          MOUNTPOINT('/u')         */

MOUNT    FILESYSTEM('CUST.ZFS.U')
         TYPE(ZFS)
         MODE(RDWR) NOAUTOMOVE
         MOUNTPOINT('/u')
```

Copying the existing /tmp and /u directories

Finally, you must copy the existing /tmp and /u directories to contain the new /tmp and /u file system, and ensure that everyone has proper access permission to the new file systems.

The z/OS UNIX command line can be accessed with the TSO OMVS command, and then you can enter these commands. Be sure to verify that each command works correctly before you proceed to the next command.

```
cd /
mkdir /tempmnt
/usr/sbin/mount –f cust.zfs.tmp /tempmnt
cp -r /tmp/ /tempmnt
```

```
ls /tempmnt
chmod 777 /tempmnt
/usr/sbin/unmount /tempmnt
/usr/sbin/mount -f cust.zfs.u /tempmnt
cp -r  /u/ /tempmnt
ls /tempmnt                 # to verify that the copy worked
chmod 777 /tempmnt
/usr/sbin/unmount /tempmnt
rm -r /tempmnt
```

Creating a loadparm for customizations

You might want to isolate your customizations into a new loadparm. For example, you can create an RC loadparm that uses your new zFS files for `/tmp` and `/u`.

- Instead of modifying `BPXPRM00`, copy it and rename it to `BPXPRMRC`, and modify `BPXPRMRC`.
- Copy the `IEASYSxx` member that currently starts the subsystems you want to start to `USER.PARMLIB`. Rename it to `IEASYSRC` to match the two character loadparm of `BPXPRMRC`. Update the **`OMVS=`** parameter to replace the existing two character loadparm with `RC`.
- Make a copy of the existing `LOADxx` member in `SYS1.IPLPARM` that currently starts the subsystems you want to start. Rename it to `LOADRC`, and store it back into `SYS1.IPLPARM`. Modify the **`SYSPARM`** value to `RC`.

Shutting down the system and re-IPLing

Shut down the system and perform an IPL. Use your loadparm that points to the modified BPXPRMxx, or your newly created load parm.

When the system restarts, you can enter the z/OS UNIX command line and enter the **`df -k`** command to ensure that your file systems are being used and mounted correctly.

After the shutdown and re-IPL is complete, you have larger zFS files for `/tmp` and `/u`.

Setting up TCP/IP

z/OS running on z Systems Development and Test Environment can communicate with your network by using TCP/IP. You can use standard 3270 terminal emulators, FTP, Developer for z Systems, and other services to move data to and from your z/OS system.

Since mainframes are confined to data centers, TCP/IP on z/OS does not act as a DHCP client. It does not automatically configure itself to a TCP/IP address supplied by the network. Therefore, it is necessary to configure a few settings to get TCP/IP to communicate with the network. Several methods of configuring TCP/IP are described in the zPDT Guide and Reference.

The subtopics in this section of the IBM Knowledge Center show an example of setting up the method referred to as Scenario 4 in Chapter 7, “LANs” in the *zPDT Guide and Reference*. This method allows the z/OS system and the Linux system to communicate with your network, each with their own IP address and with the ability to share the Ethernet adapter. Because each system has its own IP address, both systems can simultaneously use functions such as FTP with minimal configuration change. It also eliminates the need to create firewall *holes* for your z/OS ports or provide Network Address Translation functions in your firewall.

This IBM Knowledge Center shows the changes that you must make to the TCP/IP data sets in your z/OS distribution to gain that TCP/IP network connectivity. In the examples, it is assumed existing TCP/IP parmlib members and procedures are copied as new members into your `USER.*` data sets, and modified. By using this method, the parmlib members and procedures of your original z/OS distribution are unmodified. Alternatively, you can change the existing TCP/IP procedures and parmlib members without creating new members.

TCP/IP and LAN configurations are site-dependent. The exact steps that are outlined here might not work at your site because of local network configuration, firewalls, Linux dependencies, or hardware restrictions. You might need the services of a network administrator to configure z/OS within your network.

Setting up Linux routing

Before you configure TCP/IP, obtain a static IP address for z/OS. The z/OS IP address must be within the same subnet as your Linux machine. It does not matter if the Linux machine has a DHCP or static IP address if both the z/OS and Linux address are in the same subnet. The z/OS environment is configured to use both the static IP address you obtained and an address of 10.1.1.2. The address of 10.1.1.2 is used to communicate with the Linux machine and cannot be seen by other machines on your network. The following examples show how to configure z/OS so that the external network connects to the machine by address 9.12.200.20, and Linux connects to z/OS by address 10.1.1.2. z/OS can connect to the Linux machine that is using the address 10.1.1.1. The connection between z/OS and Linux is called a tunnel.

Note: You can use the `--tunnel_ip` parameter in the `awsosa` stanza to define a local address other than 10.1.1.1. For example, if you define the local address as 10.1.2.1, then define the z/OS environment as 10.1.2.2.

If you use the `create_devmap.pl` program that is described in Sample program to create device map, and you have a z/OS host name that can be resolved to your static z/OS IP address by Linux, such as through a connected Domain Name Server or static configuration, you can add the **`-h hostname`** parameter to the invocation of the program. This parameter causes `create_devmap.pl` to generate file contents that are tailored to your network. It generates the TCPIP.DATA and TCPIP.PROFILE statements by using the z/OS IP address that corresponds to the entered `hostname`, and dynamically finds the name server addresses based on the Linux ethernet adapter that is found by using `find_io`. It also generates sample VTAMLST members to define two OSAs; a tunnel OSA and an OSA to be used with one of the Linux ethernet adapters. The contents are shown as comments in the generated device map and are examples only. Because of the variability and complexity of network configuration, the samples that are shown in the generated device map might need modification to work in your environment.

This example assumes that you have the following OSA definitions in your devmap to define the tunnel OSA and one ethernet OSA to be used by TCP/IP:

```
[manager]          # define network adapter (OSA) for communication with Linux
name awsosa 0024 --path=A0 --pathtype=OSD --tunnel_intf=y   # QDIO mode
device 400 osa osa
device 401 osa osa
device 402 osa osa
[manager]          # define network adapter (OSA) for communication with network
name awsosa 22 --path=F0 --pathtype=OSD    # QDIO mode
device 404 osa osa
device 405 osa osa
device 406 osa osa
```

Modifying TCPPARMS files

Learn how to modify two TCPPARMS files to establish TCP/IP connectivity by using an IP address that is defined for z/OS: `PROFILE.TCPIP` and `TCPIP.DATA`.

PROFILE.TCPIP

Copy your existing TCP/IP profile (TCPIP.PROFILE) to USER.TCPPARMS and modify the DEVICE, LINK, HOME, and route definitions to use the OSAs defined in your devmap, with the addresses and netmask information that is correct for your network.

For example, given a z/OS IP address of 9.12.200.20, and a netmask of 255.255.255.0, a TCP/IP PROFILE member might look like the following example.

Note: Comments and other statements were removed for brevity. Also, the IP address of the Linux system is not needed here. The gateway address usually ends in `.1` or `.0`, but that might be different on your network.

This example also includes the definitions for the tunnel, the 10.1.1.2 address, which is used to communicate with z/OS from the Linux machine.

```
ARPAGE 5
DATASETPREFIX TCPIP
AUTOLOG 5
    FTPD JOBNAME FTPD1   ; FTP Server
    PORTMAP              ; Portmap Server
ENDAUTOLOG
PORT
           ((( ports removed for brevity )))
SACONFIG DISABLED
  DEVICE PORTA MPCIPA
  LINK ETH1  IPAQENET PORTA
  HOME 10.1.1.2 ETH1

  DEVICE PORTB MPCIPA
  LINK ETH2  IPAQENET PORTB
  HOME 9.12.200.20 ETH2

 BEGINROUTES
 ROUTE 10.0.0.0       255.0.0.0       =         ETH1   MTU 1492
 ROUTE 9.12.200.0     255.255.255.0   =         ETH2   MTU 1492
 ROUTE DEFAULT        9.12.200.1                ETH2   MTU 1492
 ENDROUTES
ITRACE OFF
IPCONFIG NODATAGRAMFWD
UDPCONFIG RESTRICTLOWPORTS
TCPCONFIG RESTRICTLOWPORTS
START PORTA
START PORTB
```

Copy ADCD.Z112S.VTAMLST(OSATRL2) to USER.VTAMLST(OSATRL2) and remove any comments within it so that it looks like the next example.

The device name in the TCP/IP profile member must match the port names that are specified in USER.VTAMLST(OSATRL2). In this example, these port names are PORTA and PORTB. Also, verify that your devmap (see "Defining the device map") correctly defines the device addresses in the READ, WRITE, and DATAPATH statements of USER.VTAMLST(OSATRL2).

```
OSATRL1 VBUILD TYPE=TRL
OSATRL1E TRLE LNCTL=MPC,READ=(0400),WRITE=(0401),DATAPATH=(0402),      X
               PORTNAME=PORTA,                                         X
```

```
               MPCLEVEL=QDIO
OSATRL2E TRLE LNCTL=MPC,READ=(0404),WRITE=(0405),DATAPATH=(0406),      X
               PORTNAME=PORTB,                                         X
               MPCLEVEL=QDIO
```

To activate this configuration, copy ADCD.Z112S.VTAMLST(ATCCON00) to USER.VTAMLST(ATCCON00) and change the word OSATRL1 to OSATRL2.

TCPIP.DATA

Copy your existing TCPIP.DATA data set to `USER.TCPPARMS` and set the `HOSTNAME`, `DOMAINORIGIN`, and `NSINTERADDR` values

, as in this example:

```
TCPIPJOBNAME TCPIP
HOSTNAME RDT900
DOMAINORIGIN  RTP.IBM.COM
DATASETPREFIX TCPIP
 NSINTERADDR  9.0.0.1
 NSINTERADDR  9.0.0.11
RESOLVEVIA UDP
LOOKUP LOCAL DNS
RESOLVERTIMEOUT 10
RESOLVERUDPRETRIES 1
ALWAYSWTO NO
```

If you choose a HOSTNAME or DOMAINORIGIN arbitrarily, be sure that the DOMAINORIGIN is not a real domain name and that the combination of the HOSTNAME and DOMAINORIGIN does not constitute an existing DNS name. Use the Linux `ping` or `nslookup` commands to ensure that your choice of names is not found by your DNS server. Identifying your computer as another computer, or as a member of an existing but incorrect domain, can cause problems that are difficult to diagnose, such as timeouts, pauses, and connection failures in many areas, including 3270 connections and z Systems Development and Test Environment. Some systems, including components of z Systems Development and Test Environment, require that z/OS can locate itself by name.

Ensure that the `NSINTERADDR` parameter correctly identifies a DNS server. Incorrect name server specifications can also cause long delays and errors that are difficult to diagnose. Do not specify `NSINTERADDR` values for stand-alone systems.

If you cannot use a Domain Name Server (DNS) to resolve IP addresses of other systems or of the z/OS system, you can create a local hosts file and refer to it with a GLOBALIPNODES statement. The setup of this file is described in detail in *TCP/IP implementation volume 1: Base functions* (SG24-7798).

Modifying TCP/IP Procedures to point to `USER.TCPPARMS`

An easy way to find Procedures that reference the TCPPARMS data sets by using ISPF is to display a member list of each `proclib dataset` in your active `MSTJCLxx parmlib` member (typically MSTJCL00), and then type these commands:

```
SRCHFOR TCPPARMS
SORT PROMPT
```

Copy any existing procedures that point to `TCPIP.DATA` data sets to `USER.PROCLIB`. Procedures that typically point to `TCPIP.DATA` include, but are not limited to,

```
FTPD
PORTMAP
TCPIP
TN3270
RESOLVER
```

Modify each member to change references from the existing TCPPARMS data set to USER.TCPPARMS for any members that you duplicated in USER.TCPPARMS. Do not change the member names except in the TCP/IP procedure as noted. For example, change the line in FTPD from //SYSTCPD DD DISP=SHR,DSN=CUST.TCPPARMS(TCPDATA) to:

```
//*SYSTCPD DD DISP=SHR,DSN=CUST.TCPPARMS(TCPDATA)
//SYSTCPD DD DISP=SHR,DSN=USER.TCPPARMS(TCPDATA)
```

Remember a modified TCP/IP profile member name was placed in USER.TCPPARMS, so the PROFILE DD statement is similar to this example:

```
//PROFILE  DD DISP=SHR,DSN=USER.TCPPARMS(PROFILE)
```

If you prefer to set up system-wide use of a common set of TCP/IP settings, set up the RESOLVER procedure to point to your TCP/IP definitions. Create member USER.TCPPARMS(RESOLVER) to contain the following statements.

```
GLOBALTCPIPDATA('USER.TCPPARMS(TCPDATA)')
GLOBALIPNODES('USER.TCPPARMS(IPNODES)')
COMMONSEARCH
```

Create member USER.TCPPARMS(IPNODES) to contain the following statements similar to the following but customized with your network names and address. The format of each entry in the IPNODES file is IP address followed by one or more host names. The order of the entries is not significant.

```
# This first entry is the ip address of the z/OS host with its associated hostnames
# In this example, two hostnames are configured; a shortened alias and a fully
     qualified hostname.
# Only one hostname is required, typically the fully qualified name
#
9.12.200.0   ZDT ZDT.IBM.COM
#
# This entry is the ip address of the OSA tunnel to the Linux host.
# The hostname should be the hostname of the Linux operating system.
# Multiple hostnames may be configured if needed
#
10.1.1.1     LINUX
#
# Typically, 127.0.0.1 is configured as hostname LOCALHOST
#
127.0.0.1    LOCALHOST
```

Copy your existing RESOLVER procedure to USER.PROCLIB(RESOLVER), and change the SETUP DD statement from:

```
//SETUP DD DISP=SHR,DSN=CUST.TCPPARMS(yourresolverparms),FREE=CLOSE
```

To

```
//SETUP DD DISP=SHR,DSN=USER.TCPPARMS(RESOLVER),FREE=CLOSE
```

Add the following line to the BPXPRMxx PARMLIB members you are using with z Systems Development and Test Environment. This line ensures the RESOLVER procedure that you created is the one that is started.

```
RESOLVER_PROC(RESOLVER)
```

Modifying VTAMLST members

Create a VTAMLST member to define a TRL major node that matches the OSAs you are using. Member OSATRL2 in ADCD.*.VTAMLST matches the sample devmap. If you create a new member to match your OSAs, add that member to USER.VTAMLST. To

activate this major node when VTAM® starts, add the VTAMLST member name to the VTAM configuration list (member ATCCONxx in VTAMLST) with which you start VTAM.

```
OSATRL1 VBUILD TYPE=TRL
OSATRL1E TRLE LNCTL=MPC,READ=(400),WRITE=(401),DATAPATH=(402),         X
               PORTNAME=PORTA,MPCLEVEL=QDIO
OSATRL2E TRLE LNCTL=MPC,READ=(404),WRITE=(405),DATAPATH=(406),         X
               PORTNAME=PORTB,MPCLEVEL=QDIO
```

Enabling use of USER.PROCLIB and IPLing

Ensure USER.PROCLIB (or whichever data set contains your customized proclibs) is in the concatenation of the IEFPDSI DD statement in your active MSTJCLxx parmlib member.

Copy your existing active MSTJCLxx parmlib member to USER.PARMLIB, and make changes similar to those described in this sample:

```
//MSTJCL00 JOB MSGLEVEL=(1,1),TIME=1440
// EXEC PGM=IEEMB860,DPRTY=(15,15)
//STCINRDR DD SYSOUT=(A,INTRDR)
//TSOINRDR DD SYSOUT=(A,INTRDR)
//IEFPDSI DD DSN=USER.PROCLIB,DISP=SHR << MODIFIED
// DD DSN=SYS1.PROCLIB,DISP=SHR
//SYSUADS DD DSN=SYS1.UADS,DISP=SHR
//SYSLBC DD DSN=SYS1.BRODCAST,DISP=SHR
```

Ensure USER.PARMLIB is in the PARMLIB statement concatenation of your active LOADxx member.

Also ensure USER.PROCLIB is in the concatenation of the proclibs in your procedure that starts JES. For JES2, that is the PROC00 DD statement. For JES3, that is the IATPLBST DD statement. Copy your active JES procedure into USER.PROCLIB, and make these changes.

```
//JES2      PROC MEMBER=JES2PARM,ALTMEM=JES2BACK
//IEFPROC   EXEC PGM=HASJES20,DPRTY=(15,15),TIME=1440,PERFORM=9
//ALTPARM   DD   DSN=CUST.PARMLIB(&ALTMEM),DISP=SHR
//HASPPARM  DD   DSN=CUST.&SYSVER..PARMLIB(&MEMBER),DISP=SHR
//PROC00    DD   DSN=USER.PROCLIB,DISP=SHR   << MODIFIED
//          DD   DSN=CEE.SCEEPROC,DISP=SHR
//          DD   DSN=CSQ710.SCSQPROC,DISP=SHR
//          DD   DSN=IOE.SIOEPROC,DISP=SHR
//          DD   DSN=EOY.SEOYPROC,DISP=SHR
//          DD   DSN=HLA.SASMSAM1,DISP=SHR
//          DD   DSN=CBC.SCCNPRC,DISP=SHR
//          DD   DSN=SYS1.PROCLIB,DISP=SHR
//HASPLIST  DD   DDNAME=IEFRDER
```

Perform an IPL of the system and use the loadparm that you want or your newly created loadparm, whichever is appropriate, to verify that your changes are working.

Optional tasks

Learn about optional configuration tasks for z Systems Development and Test Environment.

Logging on to TSO from emulators not native to the host Linux

After TCP/IP connectivity is established, you can use your favorite 3270 emulator software, such as IBM Personal Communications Manager (PCOMM), to connect to the non-SNA (coax) 3270 device emulator that is provided by z Systems Development and Test Environment. When you connect from outside the Linux system that is hosting z Systems Development and Test Environment, use the Linux TCPIP address and 3270 as port. (The actual port number is defined in the device map.)

Creating TSO user IDs

TSO user IDs are created through a series of commands. Typically, you want to use user IDs that have OMVS segments and an associated z/OS UNIX user directory when you are running Developer for z Systems.

The example commands that are shown here create a user ID in the RDZUSERS group with an OMVS segment. Replace #*userid*, #*name*, and #*password* with appropriate values, and do not remove the quotation marks in the commands.

From a CLIST, REXX exec, or TSO command line, enter the following commands. The commands create the user ID, provide an OMVS segment, and assign an account number, default log-on procedure, and region size. They also protect data sets with a high-level qualifier that belongs to the user from being accessed by other users. Finally, the commands create an alias in the master catalog to indicate that the users data sets are cataloged in the user catalog on volume S1CUST. Be sure that the substitution in the `HOME( )` and `PROGRAM()` parameters are in lowercase.

```
ADDGROUP RDZUSERS OMVS(AUTOGID)
ADDUSER #userid DFLTGRP(RDZUSERS) NAME('#name') PASSWORD(#password)
ALTUSER #userid OMVS(HOME(/u/#userid) PROGRAM(/bin/sh) AUTOUID)
ALTUSER #userid TSO(ACCTNUM(ACCT#) PROC(TSOLOGON) SIZE(4096) COMMAND('ISPF'))
ADDSD '#userid.*' UACC(NONE)
PERMIT ACCT# CLASS(ACCTNUM) ID(#userid) ACCESS(READ)
DEFINE ALIAS (NAME('#userid') RELATE('USERCAT.S1CUST'))
```

The new user's z/OS UNIX directory must be created. From a z/OS UNIX command line, type the following commands. You can access z/OS UNIX by typing `TSO OMVS` on the command line of any ISPF screen. You exit z/OS UNIX with the `exit` command. Again, replace `#userid` with the name of the new user ID in lowercase.

```
mkdir /u/#userid
chown #userid:RDZUSERS /u/#userid
```

Defining a new logon procedure

A common customization in z/OS systems is to alter the logon procedure that TSO users use. Do not alter the ISPFPROC logon procedure because errors might prevent you from being able to fix problems later on.

Create `USER.PROCLIB(TSOLOGON)` based on ISPFPROC, and make your modifications to TSOLOGON instead of ISPFPROC. Be sure to change the identifier on the EXEC from ISPFPROC to TSOLOGON and to override the default VOLSER used by the ISPFCL CLIST.

```
//TSOLOGON EXEC PGM=IKJEFT01,REGION=0M,DYNAMNBR=175,
//            PARM='%ISPFCL VOL(S1CFG1)'
```

To allow all users to use the TSOLOGON procedure, enter the following TSO commands.

```
RDEFINE TSOPROC TSOLOGON UACC(READ)
SETROPTS RACLIST(TSOPROC) REFRESH
```

Altering system startup and shutdown scripts

Most z/OS distributions have a shutdown script that contains a series of commands to shut down all active subsystems. Your system administrator knows the name and location of these scripts.

You might want to make these changes to these scripts.

The shutdown scripts stop OMVS automatically. Explicitly stopping ZFS can expedite the shutdown process, but requires a response to a prompt on the operator console. If you want to stop ZFS before you shut down OMVS, add

```
F OMVS,STOPPFS=ZFS
```

Before

```
F OMVS,SHUTDOWN
```

Some startup and shutdown scripts have `PAUSE` commands that you might find are too long or too short for your system. Many can be safely changed to pause for a shorter amount of time.

You can change startup scripts the same way. Other changes might include *not* starting particular subsystems, changing pause times, and so forth.

Defining JES NJE connectivity

Your z/OS distribution can be a stand-alone system with no connection to other z/OS systems. However, you might want to connect it to one or more of your z/OS systems to transfer data to customize and use the z Systems Development and Test Environment system.

Since z/OS 1.7, JES supports NJE over TCP/IP, which makes setting up a connection between two systems an easy task. The following operator commands, which are run on the z Systems Development and Test Environment system, name the local system RUT0 and define a connection to M168.

```
$TNODE(Z21S),NAME=RUT0
$TLINE1,UNIT=TCP
$SLINE1
$ADDNETSRV1,SOCKET=LOCAL
$SNETSERV1
$TNODE2,NAME=M168
$ADDSOCKET(REMOTE),NETSRV=1,LINE=1,NODE=2,IPADDR=M168.RTP.IBM.COM
$SN,SOCKET=REMOTE
```

Give similar operator commands on the M168 system to complete the setup. Since this system is an existing system, the command to define the local node name was skipped. The following commands also assume that M168 does not have spare line or node definitions, so new ones (line 5 and node 20) are created.

```
$ADDLINE5,UNIT=TCP
$SLINE5
$ADDNETSRV1,SOCKET=LOCAL
$SNETSERV1
```

```
$TNJEDEF,NODENUM=20
$TNODE20,NAME=RUT0
$ADDSOCKET(REMOTE),NETSRV=1,LINE=5,NODE=20,IPADDR=RDz8500.RTP.IBM.COM
$SN,SOCKET=REMOTE
```

The z Systems Development and Test Environment system can use the existing NJE definitions on the M168 system to connect to other NJE nodes in your network. Enter the following operator commands on the z Systems Development and Test Environment system to connect to the IPO1 system (node 3) through the previously defined M168 system (node 2).

```
$TNODE3,NAME=IPO1
$ADDCONNECT,NODEA=2,NODEB=3
```

Index