Configuring

Contents

Chapter 1. Configuring an instance of z Systems Development and Test Environment

Learn how to set up z Systems™ Development and Test Environment and customize a z/OS® software distribution for development and test purposes.

These instructions are intended to allow a person with very little z/OS systems programming experience to configure the z Systems Development and Test Environment.

Defining the device map

The z Systems Development and Test Environment allows the customization of the z Systems resources available within the virtualized environment. The resources can be defined in a device map or *devmap*.

The sections and syntax of device maps are explained in detail in the "1090 Control Files" section of the zPDT Guide and Reference.

The system volumes that represent the z/OS distribution are defined in an `awsckd` stanza, and are mounted at arbitrarily chosen addresses within the range of valid addresses that are defined by the z/OS distribution's IODF. Historically in a z/OS ADCD, the xxRES1 and xxSYS1 volumes are mounted at the same addresses as those in the *zPDT Guide and Reference* (0A80 and 0A82), and address 0AA3 is always reserved for volume xxDBAR.

If you must alter the default tunnel IP addresses of 10.1.1.1 and 10.1.1.2, refer to the example of the `--tunnel_ip` parameter in the *zPDT Guide and Reference*.

The following examples, when combined, define a working device map with a sample z/OS distribution from a z Systems Development and Test Environment ADCD subscription. If you are using the coupling facility or Rational® Tokens, see Enabling the coupling facility and Setting up Rational Tokens.

Note: The processors statement specifies the number of z Systems CPs to be used in this instance. The default is one. This number must not be more than the number available on the activated USB hardware device or more than the number of CP activations made available through a license manager.

```
[system]
memory 4096m
processors 3
3270port 3270               # port number for non-SNA (coax) 3270

[manager]
name aws3274 0001           # define non-SNA (coax) 3270 terminals
device 0700 3279 3274 mstcon
device 0701 3279 3274 tso1
```

For the network adapter definitions, the following example was added. Your setup might differ, and you are encouraged to consult the sections on connectivity in the *zPDT Guide and Reference*.

```
[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
```

Since the volumes are in the /home/ibmsys1/z1090/disks/ directory, the DASD definitions might look like the following example:

```
[manager]
name awsckd 0002
device 0A80 3390 3390 /home/ibmsys1/z1090/disks/B2RES1
device 0A81 3390 3390 /home/ibmsys1/z1090/disks/B2BLZ1
device 0A82 3390 3390 /home/ibmsys1/z1090/disks/B2SYS1
device 0A83 3390 3390 /home/ibmsys1/z1090/disks/B2CFG1
device 0A84 3390 3390 /home/ibmsys1/z1090/disks/B2C511
device 0A85 3390 3390 /home/ibmsys1/z1090/disks/B2C521
device 0A86 3390 3390 /home/ibmsys1/z1090/disks/B2C531
device 0A87 3390 3390 /home/ibmsys1/z1090/disks/B2DBA2
device 0A88 3390 3390 /home/ibmsys1/z1090/disks/B2DBA1
device 0A89 3390 3390 /home/ibmsys1/z1090/disks/B2DBB1
device 0A8A 3390 3390 /home/ibmsys1/z1090/disks/B2DBB2
device 0A8B 3390 3390 /home/ibmsys1/z1090/disks/B2DIS1
device 0A8C 3390 3390 /home/ibmsys1/z1090/disks/B2DIS2
device 0A8D 3390 3390 /home/ibmsys1/z1090/disks/B2IMD1
device 0A8E 3390 3390 /home/ibmsys1/z1090/disks/B2IME1
device 0A8F 3390 3390 /home/ibmsys1/z1090/disks/B2IMU1
device 0A90 3390 3390 /home/ibmsys1/z1090/disks/B2IMU2
device 0A91 3390 3390 /home/ibmsys1/z1090/disks/B2KAN1
device 0A92 3390 3390 /home/ibmsys1/z1090/disks/B2PAGA
device 0A93 3390 3390 /home/ibmsys1/z1090/disks/B2PAGB
device 0A94 3390 3390 /home/ibmsys1/z1090/disks/B2PAGC
device 0A95 3390 3390 /home/ibmsys1/z1090/disks/B2PRD1
device 0A96 3390 3390 /home/ibmsys1/z1090/disks/B2PRD2
device 0A97 3390 3390 /home/ibmsys1/z1090/disks/B2PRD3
device 0A98 3390 3390 /home/ibmsys1/z1090/disks/B2RES2
device 0A99 3390 3390 /home/ibmsys1/z1090/disks/B2USS1
device 0A9A 3390 3390 /home/ibmsys1/z1090/disks/B2USS2
device 0A9B 3390 3390 /home/ibmsys1/z1090/disks/B2W801
device 0A9C 3390 3390 /home/ibmsys1/z1090/disks/B2W802
device 0A9D 3390 3390 /home/ibmsys1/z1090/disks/B2W803
device 0A9E 3390 3390 /home/ibmsys1/z1090/disks/B2W804
device 0A9F 3390 3390 /home/ibmsys1/z1090/disks/B2W805
device 0AA0 3390 3390 /home/ibmsys1/z1090/disks/B2W851
device 0AA1 3390 3390 /home/ibmsys1/z1090/disks/B2W852
device 0AA2 3390 3390 /home/ibmsys1/z1090/disks/B2DBAR
device 0AA3 3390 3390 /home/ibmsys1/z1090/disks/B2W853
device 0AA4 3390 3390 /home/ibmsys1/z1090/disks/B2W854
device 0AA5 3390 3390 /home/ibmsys1/z1090/disks/B2W855
device 0AA6 3390 3390 /home/ibmsys1/z1090/disks/B2W856
device 0AA7 3390 3390 /home/ibmsys1/z1090/disks/B2W857
device 0AA8 3390 3390 /home/ibmsys1/z1090/disks/SARES1
device 0AA9 3390 3390       # Available for dynamic mounts
device 0AAA 3390 3390       # Available for dynamic mounts
device 0AAB 3390 3390       # Available for dynamic mounts
```

Figure 1. Device map example

You can verify the device map with the awsckmap command. All of the disk volume images that are referenced in the device map must exist before you verify the device map. Assume that the device map file has the following name:

```
/home/ibmsys1/z/myDeviceMap
```

With the name in the preceding example, the device map can be verified with the command:

```
awsckmap /home/ibmsys1/z/myDeviceMap
```

Sample program to create device map

A sample program that is called `create_devmap.pl` is available in the `ConfigGuideSample` directory where you installed z Systems Development and Test Environment. If Perl is installed on your Linux system, you can use the `create_devmap.pl` program to generate a sample device map that is based on your current decompressed 3390 disk images, memory configuration, and available network parameters. Consider the output of `create_devmap.pl` to be a starting point from which you can create a final device map.

The syntax for the `create_devmap.pl` command is shown here:

```
perl <pathtocommand>/create_devmap.pl pathtodisks > generateddevmap
```

In the preceding example, *pathtocommand* is the location of the `create_devmap.pl` file, and *pathtodisks* is the location of your 3390 disk images. *generateddevmap* is the name of the file to contain the new device map.

If you already have a static IP address that is assigned for your virtual z/OS machine and a z/OS host name that can be resolved to that address by Linux, such as through a connected Domain Name Server or static configuration, you can add the `-h <hostname>` parameters after the `pathtodisks` parameter. Adding the parameter causes the script to attempt to generate comments that contain more accurate z/OS TCP/IP configuration samples based on your network.

```
perl <pathtocommand>/create_devmap.pl pathtodisks -h
   hostname_of_zos > generateddevmap
```

The `create_devmap.pl` program creates a `memory` line based on existing hardware and configuration of your Linux machine. Verify that the amount of memory that is requested is appropriate for your situation.

The device map that is created by `create_devmap.pl` defines OSA devices based on the first Tun/Tap and Wired CHPIDs found that use the **find_io** command, and a set of sample z/OS TCP/IP definitions that would correspond to the OSA device definitions in the generated device map. These TCP/IP configuration statements can be used as a starting point for your TCP/IP configuration, but probably requires changes to match your network. Verify that the device addresses and device names in your final VTAM® definitions, TCP/IP profile, and device map all correspond to the correct network adapter types.

The device map that is created by `create_devmap.pl` also contains 3390 device statements for files in the *pathtodisks* directory that are over 800 MB, have six-character names, and are verified to be disk images by the `alcckd` command.

Chapter 2. Starting and stopping z Systems Development and Test Environment

Learn how to start and stop z Systems Development and Test Environment.

Starting the z Systems Development and Test Environment

If you use the same directory structure that is used in these examples, and x3270 is installed on your native Linux system, you can start your z Systems Development and Test Environment system with a script similar to the example in this topic.

The script example that is used in this document is named: `/home/ibmsys1/z/runzpdt`. After you create the script, you must ensure that it is executable by entering this command: **`chmod 755 /home/ibmsys1/z/runzpdt`**.

This script is written to accept two command line switches. The `-d` switch can be used to specify a device map and the `-l` switch (the lowercase letter L) can be used to specify the load parameter. The sample `runzpdt` script is available in the `ConfigGuideSample` directory and is shown here:

```
#!/bin/bash
LOADPARM=CS
DEVMAP=myDeviceMap

#cd /home/ibmsys1/z  # Optional: the directory from which you want to run

while getopts "d:l:" opt ; do
  case $opt in
     d)
 DEVMAP=$OPTARG
  ;;
     l)
 LOADPARM=$OPTARG
 ;;
     \?)
 echo "Invalid paramater:" $OPTARG
 echo " runzpdt [-d devmap] [-l loadparm]"
 exit 1
 ;;
  esac
done

PORT=`egrep "^3270port" $DEVMAP | awk '{print $2}'`

echo "Load parm: $LOADPARM,  Devmap: $DEVMAP,  Port: $PORT"

if [ ! -e $DEVMAP ]; then
 echo "Devmap file $DEVMAP does not exist"
 exit 1
fi

echo stopping previous instance
awsstop
killall -u $(id -un) x3270
while ps -U $(id -un) |egrep "emily|aws.{3,5}" >/dev/null; do sleep 1;done

# start Rational Development & Test environment.  --clean is optional
echo awsstart $DEVMAP  # --clean
awsstart $DEVMAP # --clean

egrep "AWS[A-Z]{3}[0-9]{3}[ES]"  `ls -t1 ~/z1090/logs/log_console_* | head -n 1` 1>/dev/null
if [ $? -ne 0 ]; then
 echo
 echo "Rational Development and Test Environment started."
 echo
 /usr/z1090/bin/token
 # start x3270 for the console and one local user terminal

 nohup x3270 -model 4 mstcon@localhost:$PORT  1>/dev/null 2>/dev/null &
 nohup x3270 -model 4 tso@localhost:$PORT     1>/dev/null 2>/dev/null &

 #Perform the IPL of the system
 echo ipl a80 parm 0a82$LOADPARM
 ipl a80 parm 0a82$LOADPARM
else
 echo
 echo "No completed startup message was found."
 echo '****************************************'
 echo '*** runzpdt terminated with errors. ***'
 echo '****************************************'
fi
```

Figure 2. Startup example

The *ipl* statement contains three pieces of information. The a80 is the device address of the SYSRES volume, which is a bootable z/OS volume. The parameter string 0A82xx specifies the 4-digit device address of the IODF volume (which holds

IPL configuration files) and the LOADPARM, the suffix that identifies the `LOADxx` member that is used to start z/OS. In the script, the LOADPARM is a variable that you can change by using the parameter `–l xx` when you start the script. (The `-l` is a lowercase "L".)

The first time you start a new z/OS distribution, start z Systems Development and Test Environment with a loadparm that does a cold start, and does not start additional subsystems. After you verify basic z/OS capabilities, you can then customize the system by using the instructions in Configuring a working z/OS system and create and start different loadparms as needed. As with any z/OS system, warm starts are less disruptive and preserve the JES job spool. Use warm starts when possible.

IPLing z Systems Development and Test Environment from a remote emulated terminal for the system console

If your Linux image with the installed z Systems Development and Test Environment does not have a user interface, you cannot use an emulated terminal product such as X3270 to run your z/OS console. Therefore, you must install the emulated terminal software on a remote system, and you must be dialed to the Linux system that is running z Systems Development and Test Environment before you enter the **`ipl`** command.

If the remote emulated terminal product is a 3270 emulator, then the only difference between remote administration and local administration of z Systems Development and Test Environment are the commands that you enter between the `awsstart` and `ipl` commands. For remote administration, rather than starting x3270 or another local 3270 emulator, instead, start the 3270 emulator on the remote system and connect to the Linux IP address and zPDT® 3270port (usually 3270). From this point forward, the remote 3270 emulator becomes your system console. Any wanted TSO sessions can be connected from remote systems as well using the defined x3270port.

If you want to use the supplied `runzpdt` script to start z Systems Development and Test Environment, you must modify it to perform remote system console administration. Any changes to `runzpdt` depend on what method of starting z Systems Development and Test Environment and automation you currently use. You can customize these alternative modifications to fit your environment.

- Remove the final stanzas after the `awsstart` that start the local x3270 sessions and perform the **`ipl`**. The `ipl` command can be run as a line command after the system console is connected.
- Replace the x3270 start commands in `runzpdt` with some form of pause while you start your remote system console (either manually or with automation). These examples illustrate two pause techniques:
  - A sleep statement, such as

    ```
    sleep 1m
    ```

  - A read statement, such as

    ```
    read -p "Press [Enter] key to ipl after system console is connected."
    ```

Stopping Rational Development and Test Environment for z Systems

If possible, always shut down z/OS cleanly. Typically, shutting down cleanly begins by starting a procedure that shuts down all active subsystems.

Any z/OS ADCD for z Systems Development and Test Environment contains sample shutdown scripts for the systems that are available and active in that distribution. For an example of these shutdown scripts, see Altering system startup and shutdown scripts. This example shows the type of commands you can use in a shutdown script.

```
/*-------------------------------------------------------------------*/
/* WARN TSO USERS TO LOGOFF                                          */
/*-------------------------------------------------------------------*/
F TSO,USERMAX=0        /* DON'T ALLOW ANYONE ELSE TO LOGON RIGHT NOW! */
SEND 'PLEASE LOGOFF - THE SYSTEM WILL BE IPLED IN 2 MINUTES!!',ALL,NOW
PAUSE 10
/*-------------------------------------------------------------------*/
/* ISSUE STOP COMMANDS FOR ALL TASKS THAT WILL TAKE THEM.  FOR THOSE  */
/* THAT WON'T TAKE THEM, SIMPLY ISSUE CANCEL COMMANDS.               */
/*-------------------------------------------------------------------*/
/*-------------------------------------------------------------------*/
/* ISSUE MODIFY COMMANDS FOR THOSE TASKS THAT USE THEM INSTEAD OF STOP*/
/*-------------------------------------------------------------------*/
/*-------------------------------------------------------------------*/
/* TRY TO WAIT LONG ENOUGH FOR ALL TO COME COMPLETELY DOWN            */
/*-------------------------------------------------------------------*/
P TSO
C INETD4
P LLA
SETRRS SHUTDOWN
P VLF
MODIFY DLF,MODE=Q
P DLF
P HZSPROC
P TN3270
P TCPIP
P SDSF
PAUSE 20
Z NET,QUICK
PAUSE 10
F OMVS,SHUTDOWN
```

During shutdown, you might have to respond to z/OS console messages, such as when IMS™, TSO, or z/OS UNIX are stopped. You can see what programs are still running by entering the `D J,L` console command. Ensure VTAM and all subsystems end.

After all systems are stopped, stop your JES system. After JES ends, z Systems operation can be stopped. Enter a **`QUIESCE`** command from the MVS™ console to ensure that there is no more activity to the z/OS volume image files. The zPDT system can then be stopped with this command in the Linux window:

```
$ awsstop
```

This command produces several messages. It might be necessary to press **Enter** to obtain the Linux prompt. Any 3270 windows can be closed.

Index