

# IBM System z Personal Development Tool Volume 3 Additional Topics

System z Development Tool

Full z/OS usage

Linux base



Bill Ogden

**Redbooks**





International Technical Support Organization

**IBM System z Personal Development Tool: Volume 3  
Additional Topics**

December 2010

**Note:** Before using this information and the product it supports, read the information in “Notices” on page vii.

**Third Edition (December 2010)**

This edition applies to the IBM 1090 (also known as zPDT) release that is available at the time of publication.

**© Copyright International Business Machines Corporation 2009, 2010. All rights reserved.**

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contents

<b>Notices</b> .....	vii
Trademarks .....	viii
<b>Preface</b> .....	ix
The author .....	ix
Now you can become a published author, too! .....	ix
Comments welcome .....	x
Stay connected to IBM Redbooks .....	x
<b>Chapter 1. General usage</b> .....	1
1.1 Token dates and times .....	2
1.2 CPs, CPUs, threads, and tokens .....	2
1.3 VM active wait and z/OS spinloops .....	4
1.4 CKD versioning .....	5
1.5 1090 messages .....	6
1.6 TCP/UDP port for token .....	6
1.7 OSA CHPIDs .....	6
1.8 Dual boot .....	7
1.9 Remote operation .....	9
1.10 x3270 cursor position .....	9
1.11 Devices, memory, msgmni, ulimit .....	9
1.12 Linux ftp and telnet .....	10
1.13 Startup scripts .....	10
<b>Chapter 2. Tapes (SCSI and awstape)</b> .....	13
2.1 SCSI tape drives .....	14
2.1.1 The awsscsi device manager .....	14
2.2 SCSI utilities .....	17
2.3 awstape utilities .....	17
<b>Chapter 3. z/OS notes</b> .....	19
3.1 z/OS CP and memory display .....	20
3.2 z/OS spin loop timeouts .....	20
3.3 Larger panel .....	20
3.4 z/OS disk space .....	20
3.5 Java and WebSphere Application Server startup .....	21
3.6 Stand-alone z/OS dump .....	24
3.6.1 Generating a stand-alone dump program .....	25
3.6.2 Taking a stand-alone z/OS dump .....	25
3.7 Moving 3390 volumes .....	26
3.7.1 Create source dump .....	28
3.7.2 Send dump to Linux .....	29
3.7.3 Receive dump .....	29
3.7.4 Stand-alone restore .....	30
3.8 IODF Changes with zPDT .....	34
3.9 Local printing .....	36
3.9.1 Setup .....	37
3.9.2 Operational technique .....	39
3.10 Enabling TSO users for OMVS .....	39

3.11	SYS1.LOGREC full	40
3.12	Lost MVS console	41
3.13	Unable to start ISPF	42
3.14	Health Checker	42
3.15	Reply needed to stop zFS	43
3.16	RMF-III	43
3.17	Compressing PARMLIB	43
3.18	SVC dumps for IRARMCPU	43
3.19	Burning 3390 volumes on CD	44
3.20	Delete logstreams	44
3.21	Installing a DEMOpkg	44
3.22	Disabled waits	46
<b>Chapter 4. z/VM notes</b>		<b>51</b>
4.1	Installing z/VM 6.1	52
4.2	A prepackaged z/VM 5.4 system	53
4.3	Installing the AD-CD z/VM 5.4 DVD	54
4.3.1	1090 devmap	54
4.3.2	Initial IPL	54
4.3.3	Initial usage	57
4.3.4	AUTOLOG1 functions	59
4.4	Multiple guest z/OS users	60
4.4.1	Running z/OS under z/VM	64
4.5	<i>zIIPs and zAAPs</i>	65
4.6	Paging	65
<b>Chapter 5. z/VSE notes</b>		<b>67</b>
<b>Chapter 6. Multiple zPDT instances</b>		<b>71</b>
6.1	Multiple instances or guests	72
6.2	Multiple guests in one instance	72
6.3	Independent instances	73
6.4	Instances with shared I/O	75
6.5	Additional shared functions	79
<b>Chapter 7. Using awscmd</b>		<b>81</b>
7.1	Sample z/VM script	82
7.2	z/OS usage	83
7.2.1	Sample z/OS program for awscmd	84
<b>Chapter 8. Problem handling</b>		<b>91</b>
8.1	Problems starting 1090 operation	92
8.2	Problems during 1090 operation	92
8.3	Core images	95
8.4	Emulated volume problems	95
8.5	Linux monitoring	96
<b>Chapter 9. Linux for System z</b>		<b>99</b>
9.1	SUSE Linux Enterprise Server	100
9.1.1	Comments	103
9.2	Red Hat Enterprise Linux for System z under z/VM	104
<b>Chapter 10. LAN notes</b>		<b>113</b>
10.1	Recent changes	114
10.2	Non-QDIO operation	114

10.2.1 More complete example . . . . .	115
10.3 Local routers and DHCP . . . . .	118
10.4 Shared Ethernet adapters . . . . .	119
10.5 Base Linux LAN notes . . . . .	121
<b>Chapter 11. DASD volume migration . . . . .</b>	<b>123</b>
11.1 Warnings . . . . .	124
11.2 Operational characteristics of the migration utility . . . . .	124
11.3 Installation of the migration utility for z/OS . . . . .	125
11.3.1 Server installation . . . . .	126
11.3.2 RACF requirements . . . . .	127
11.4 Operation of the server under z/OS . . . . .	128
11.5 Installation of the server under z/VM . . . . .	128
11.6 Operation of server under z/VM . . . . .	129
11.7 The client commands . . . . .	129
11.8 Additional notes . . . . .	130
<b>Chapter 12. Channel-to-channel . . . . .</b>	<b>133</b>
12.1 z/OS usage example . . . . .	135
12.2 Multiple instances and z/VM . . . . .	137
12.2.1 Devmaps . . . . .	137
<b>Chapter 13. Cryptographic adapter . . . . .</b>	<b>141</b>
13.1 Background information . . . . .	142
13.2 Devmap specification . . . . .	142
13.3 Initial ICSF startup . . . . .	143
13.4 Operational notes . . . . .	146
13.4.1 Multiple zPDT instances . . . . .	147
13.4.2 Coprocessor control commands . . . . .	148
13.4.3 New z/OS releases . . . . .	149
13.4.4 Programming with ICSF . . . . .	149
13.4.5 z/VM usage . . . . .	151
<b>Related publications . . . . .</b>	<b>153</b>
IBM Redbooks . . . . .	153
Other publications . . . . .	153
How to get Redbooks . . . . .	153
Help from IBM . . . . .	153
<b>Index . . . . .</b>	<b>155</b>



# 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.

# Trademarks

IBM, the IBM logo, and [ibm.com](http://www.ibm.com) are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

GDDM®	Redbooks®	z/OS®
IBM®	Redbooks (logo)  ®	z/VM®
OS/390®	System z®	z/VSE™
PR/SM™	VTAM®	
RACF®	z/Architecture®	

The following terms are trademarks of other companies:

AppArmor, SUSE, the Novell logo, and the N logo are registered trademarks of Novell, Inc. in the United States and other countries.

Red Hat, and the Shadowman logo are trademarks or registered trademarks of Red Hat, Inc. in the U.S. and other countries.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

# Preface

This IBM® Redbooks publication introduces the IBM System z® Personal Development Tool (zPDT), which runs on an underlying Linux® system based on an Intel® processor. zPDT provides a System z system on a PC capable of running current System z operating systems, including emulation of selected System z I/O devices and control units. It is intended as a development, demonstration, and learning platform; it is not designed as a production system.

This book, discussing more advanced topics, is the last of three volumes. The first volume introduces zPDT and provides reference material for zPDT commands and device managers. The second volume describes the installation of zPDT (including the underlying Linux, and a particular z/OS® distribution) and basic usage patterns. The third volume discusses more advanced topics that may not interest all zPDT users.

The systems discussed in these volumes are complex, with elements of Linux (for the underlying PC machine), z/Architecture® (for the core zPDT elements), System z I/O functions (for emulated I/O devices), and z/OS (providing the System z application interface), and possibly with other System z operating systems. We assume the reader is familiar with the general concepts and terminology of System z hardware and software elements and with basic PC Linux characteristics.

## The author

This series of IBM Redbook publications was produced by the zPDT development team, with assistance from many other people.

**Bill Ogden** is a retired Senior Technical Staff Member at the International Technical Support Organization, Poughkeepsie. He enjoys working with new mainframe users and entry-level systems.

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

**Keith VanBenschoten**, IBM Poughkeepsie, was very helpful in establishing installation and startup processes for the 1090 and in providing test systems.

**Theodore Bohizic**, IBM Poughkeepsie, helped us understand command, design, and internal details.

**Richard Brandle**, IBM Dallas, helped with much of the practical usage information incorporated in this book.

**Kelly Ryan**, IBM Poughkeepsie, provided importance directions for determining user levels and associated support requirements.

## Now you can become a published author, too!

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You

will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

[ibm.com/redbooks/residencies.html](http://ibm.com/redbooks/residencies.html)

## Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- ▶ Use the online **Contact us** review Redbooks form found at:

[ibm.com/redbooks](http://ibm.com/redbooks)

- ▶ Send your comments in an e-mail to:

[redbooks@us.ibm.com](mailto: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

## Stay connected to IBM Redbooks

- ▶ Find us on Facebook:

<http://www.facebook.com/IBMRedbooks>

- ▶ Follow us on twitter:

<http://twitter.com/ibmredbooks>

- ▶ Look for us on LinkedIn:

<http://www.linkedin.com/groups?home=&gid=2130806>

- ▶ Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>

- ▶ Stay current on recent Redbooks publications with RSS Feeds:

<http://www.redbooks.ibm.com/rss.html>



# General usage

This chapter contains a variety of topics, in no particular order. This information is not required for basic 1090 operation but it is helpful for better understanding the 1090 and for more advanced uses.

## 1.1 Token dates and times

The 1090 token remembers the latest date and time that it sees from the underlying Linux system. It uses the PC “hardware clock” for this purpose, without any time zone adjustment. If the PC clock is set to UTC, then subsequent adjustments for time zones and daylight savings time do not affect the token.

The token must never see the date or time move backwards. If this happens, a *time cheat* message is produced and the 1090 does not start.

For example, if you set the PC date ahead several months (perhaps to test an expiration function in the application you are developing) and you use the 1090 with this advanced date, you cannot then return to the correct date and use the 1090 token.<sup>1</sup> If you inadvertently used an incorrect (future) date with the 1090, and you now find your 1090 unusable with the correct date, you should contact the business partner that furnished the 1090.

If you must temporarily change the PC clock (when not using zPDT), remove the 1090 token before doing this and reset the clock to the current time before connecting the 1090 token again.

If you move a token among multiple PCs, you should take care that the hardware time-of-day clocks are close to each other on all the machines.

## 1.2 CPs, CPUs, threads, and tokens

The L01 model of the 1090 provides one CP. The L03 model provides up to three CPs. Each can be used as a normal CP or as a zIIP, zAAP, or IFL. The model number determines the maximum number of CPs or zIIPs or zAAPs or IFLs for the 1090—and the model number is determined by the USB hardware key (token). This discussion refers to CPs, but also applies to zIIPs, zAAPs, and IFLs. The CPs of an L03 may be used in a single instance (that is, three CPs available to a single copy of z/OS) or in three instances (each with a single CP) or some combination of these.<sup>2</sup> The System z CPs (or zIIPs, or zAAPs, or IFLs) are the logical product of the 1090 system.

There is not a one-to-one correspondence between logical CPs and underlying CPUs on the base hardware. Each CP is represented by a Linux process on the underlying system, and this process has multiple *threads*.<sup>3</sup> One thread is used for the primary operation of the CP. Other threads may help prepare System z instructions for the primary thread. If multiple CPUs are available on the underlying system, then multiple threads of a CP process may run in parallel.

Stated another way, multiple PC cores might contribute to the operation of a single CP, resulting in CP performance better than what could be produced by a single PC processor. Additional PC processors are also used for other processes, such as I/O for the 1090, x3270 sessions, and the normal Linux background tasks. Considering a W500 mobile computer with two processors and a 1090 L01 license, one CPU is consumed for the CP process.<sup>4</sup> The second processor is partly used for I/O and so forth, and partly used to help prepare

<sup>1</sup> The 1090 developers are aware there is a need for a method to test application expiration routines.

<sup>2</sup> Multiple instances are something like multiple LPARs, but without many of the auxiliary facilities of LPARs.

<sup>3</sup> A *thread* is a dispatchable unit for Linux. It is something like a *task* for z/OS.

<sup>4</sup> A particular CPU is not dedicated to CP operation. CPUs are subject to normal Linux dispatching. The CPU dispatched to the primary CP thread might change due to Linux dispatching. In trivial cases on a dual processor T60, we have noticed that Linux appears to switch which CPU is dispatched to the primary CP thread every second or so.

instructions for the CP process. Ignoring all the Linux background processes and 1090 I/O, a PC with more processors than CPs will generally have better CP performance than a machine where the number of CPs equals the number of PC processors.<sup>5</sup>

The internal operation of the 1090 is complex and not documented. We cannot offer specific tuning information to attempt to optimize the CP/processor mix. It may be possible to construct workloads that perform worse with “extra” processors available, and it is possible to construct other workloads that perform twice as well as they would with a single PC processor. With typical workloads, over a reasonable time period, the availability of an “extra” PC processor contributes to the performance of a CP.

At this time, the availability of more than twice as many PC processors as CPs appears to provide no additional performance improvement. That is, using a four-processor PC to run one CP does not appear to offer much advantage over a two-processor PC. What little advantage might be seen is probably due to Linux background tasks using the additional processors.

When 1090 operation starts, the `awsstart` command processes the devmap. The devmap specifies the number of CPs to start.<sup>6</sup> (It defaults to one CP.) The `awsstart` program requests the specified number of licenses from the USB hardware key. A model L01 key can supply only one license.

An L02 is not twice as fast as a model L01 (on a dual processor PC) because PC resources are still needed for I/O, background Linux processes, and the use of multiple threads to prepare instructions for each CP. In very informal usage, we have seen performance improvements up to 25% in some cases.<sup>7</sup> An L03 licenses provides three CPs (or mixtures of CPs, zIIPs, zAAPs). These could be used in a variety of ways, as suggested in Figure 1-1.

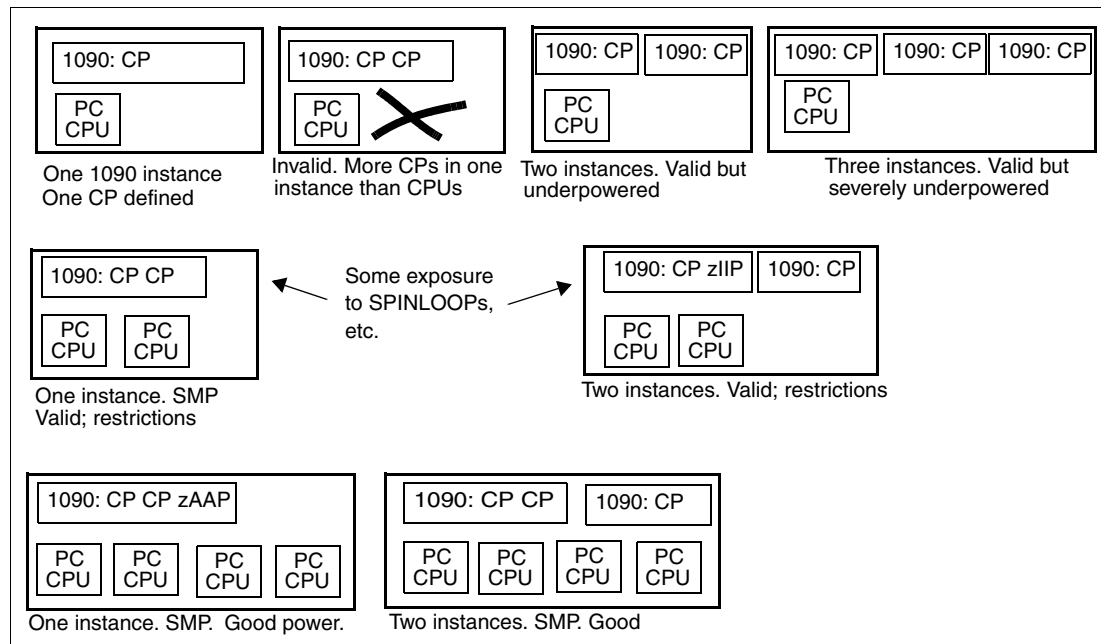


Figure 1-1 CP, CPU, and instances

The rules, implied in Figure 1-1, are these:

<sup>5</sup> A 1090 cannot have fewer CPUs than the number of CPs in the largest instance.  
<sup>6</sup> This is an overview; the internal details for processing the 1090 license are not described.  
<sup>7</sup> This was on a dual-core mobile computer.

- ▶ Your 1090 model number (L01, L02, or L03) must be greater than or equal to the number of (CPs + zIIPs + zAAPs + IFLs) in your total configuration.
- ▶ You cannot have more (CPs + zIIPs + zAAPs + IFLs) in a single instance than there are CPUs in your underlying processor.
- ▶ If the number of (CPs + zIIPs + zAAPs + IFLs) in your largest (or only) instance equals the number of CPUs in your underlying processor, there is a chance for spinloops to occur (or similar problems in operating systems other than z/OS). The possibility of this happening can be reduced by eliminating as many non-1090 processes in your Linux system as possible.

## 1.3 VM active wait and z/OS spinloops

The *vsi=on* option (in the [system] section of a devmap) performs two functions:

- ▶ It indicates to z/VM® that *active wait* should not be used. Active wait is used by z/VM when it is not in an LPAR environment. Active wait causes z/VM to *spin* when waiting for work, as opposed to entering a wait state until an interrupt signals new work. For the 1090 environment, active wait tends to use 100% of one of the PC processors, regardless of whether z/VM is doing any work. We expect that most 1090 users will use the *vsi=on* option when they use z/VM.
- ▶ As an additional function, this option causes z/OS to take special internal actions to more quickly resolve spinloop situations. However, the use of *vsi=on* creates a small performance degradation for z/OS.

The use of *vsi=on* with z/VM is straightforward and is sometimes known as “setting the LPAR bit” for z/VM. (The 1090 does not implement System z PR/SM™ LPAR functions, but some of the associated terminology is sometimes used with the 1090.)

The other function of the *vsi=on* option in the 1090 is useful when a user is seeing z/OS SPINLOOP error messages. (Note that we are not discussing z/OS operation under z/VM; this *vsi=on* function is not related to z/VM usage.) Such spinloop errors most often occur when the number of defined 1090 CPs is equal to the number of the underlying PC processors.

A spinloop should never occur when a single System z CP is defined for the 1090, even if there is only a single underlying PC processor. The most common occurrence of a spinloop timeout occurs when defining two CPs where there are only two underlying PC processors; this is typical on a W500 or T61 ThinkPad that has a dual-core processor element.

The performance degradation is small; informal measurements indicate it is less than 3%. In general, we recommend specifying the *vsi=on* option when you define multiple CPs (in the devmap) and the number of CPs defined is equal to the number of processors in the underlying PC.

The use of *vsi=on* may have side effects in rare situations, although we have not seen this in the most recent versions of the 1090. When this option is set, the System z operating system (usually z/OS) operates as though it were running in an LPAR and may attempt to access various LPAR functions that do not exist on the 1090. Using *vsi=off* causes the System z operating system to detect a basic<sup>8</sup> environment (no LPAR).

<sup>8</sup> Basic mode (without LPARs) no longer exists on larger System z machines, but this description should be adequate for 1090 usage.

## 1.4 CKD versioning

A function is available to allow an emulated 3390 (or 3380) volume to be “reset” to a selected point in time. This function is known as *CKD versioning*. The usage is as follows:

- ▶ A command is issued for selected emulated volumes (which might be all the emulated volumes in your configuration) to enable versioning. This must be done when zPDT is not active.<sup>9</sup>
- ▶ zPDT is started, an operating system is IPLed, and the volumes are used in the normal way.
- ▶ At a later time (when zPDT is not running), commands are issued to either commit whatever changes were made to the volumes, or restore the volumes to the exact content it had when CKD versioning was enabled for it.

A typical use might be for a demonstration or benchmark run. After the demonstration or benchmark run is completed, the volumes can be restored to their original state.

A CKD-emulated volume can be considered to have two versions. Version zero is the original state of the volume and version one is a volume that has been changed after versioning was enabled. Only one restore version is possible for changed volumes. That is, multiple generations of versions are not possible.

The commands associated with CKD versioning are:

```
$ alckd /z/WORK01 -ve      enable versioning for the indicated volume
$ alckd /z/WORK01 -vr      restore volume to original content
$ alckd /z/WORK01 -vc      commit the changes to the volume
$ alckd /z/WORK01 -vi      inquire about the versioning status of the volume
$ alckd /z/WORK01 -vd      disable versioning (if no changes have been made)
```

These examples use an emulated volume stored in /z/WORK01. You would specify the name of the Linux file containing your volume, of course.

When changes are made to a track of a version-enabled volume, the original track contents are saved at the end of the emulated volume file. Only one “original track” is saved; subsequent changes to the track simply update the track within the emulated volume. If the volume is *restored*, the original track(s) replace the changed track(s). If the changes are *committed*, the original tracks are discarded.

Restoring or committing a volume results in a volume that is not enabled for versioning. It can be enabled again with the **-ve** option.

The version-enabled status of a volume is carried over subsequent zPDT starts and stops, and subsequent operating system IPLs. The version-enabled status remains until the volume is restored or committed. The Linux file size for the emulated volume grows as additional “original tracks” are stored. In an extreme case, where every track on the volume is changed, the Linux file will grow to twice its original size. In typical cases, relatively few tracks on a volume are changed through normal usage and the Linux file growth is minor. Restoring or committing changes causes the Linux file to return to its original size.

If versioning has been enabled for a volume, but there have been no changes to the volume, then the **-vd** option can be used to disable versioning. If there have been changes to the volume (causing the versioning function to start operation) then the **-vd** option is rejected. At this point you must use **-vr** (to restore the volume to the original state) or **-vc** (to commit the changes) to disable the versioning function.

---

<sup>9</sup> zPDT can be active if the selected volumes are not in the active devmap.

The **-vi** (inquiry) option displays the current versioning state of the volume and, if versioning is active, displays the number of CKD tracks that have been versioned.

## 1.5 1090 messages

Messages issued by the 1090 have unique message numbers. The 1090 **msgInfo** command can be used to obtain more information about a message. An example of usage might be as follows:

```
$ awsckmap aprof1                (check my devmap)
AWSCHK200I Checking DEVMAP file 'aprof9' ...
AWSCHK204I Processed 204 records from DEVMAP /home/ibmsys1/aprof1
AWSCHK208I Check complete, 0 errors, 0 warnings detected.

$ msgInfo AWSCHK208I
AWSINF010I Format:
AWSINF013I     AWSCHK208I Check complete, %d error%s, %d warnings detected.
AWSINF013I
AWSINF011I Description:
AWSINF013I     The DEVMAP check is complete.
AWSINF013I
AWSINF012I Action:
AWSINF013I     Informational message only. No corrective action needed but
AWSINF013I     if errors are present the DEVMAP cannot be used to start system.
```

All message numbers are in the form of AWScccnnns, where:

ccc is the component code issuing the message.

nnn is the message number within the component.

s is the message severity (**D**ebug, **I**nformation, **W**arning, **E**rror, **S**evere, **T**erminal)

The message code specified on the **msgInfo** command can omit the AWS prefix and the severity code. For example, `msgInfo chk208` is sufficient. There is also an environment variable named `Z1090_MSG` to control message formatting. It may be set to `FULL` (the default), `CODE` (which will only print the message number and no text), `TEXT` (which prints the message text and no code) and `SHORT` (which drops the AWS prefix on the message number).

## 1.6 TCP/UDP port for token

The 1090 token device is accessed through TCP/UDP and requires a port number. A *well known* port number has been assigned for this device; this is port 9450.

## 1.7 OSA CHPIDs

All your LAN interfaces should be configured (and tested) for Linux use before attempting to use one or more interfaces for OSA Express emulation.

The Ethernet adapters used by `awsosa` are identified by the path parameter in the devmap or in the IOCDs. The path parameter is a CHPID number, and you must determine the correct CHPID number by using the `find_io` command. The `find_io` command should produce

output similar to the following when two LAN interfaces are available. In this example, the second interface was an Etherjet PC Card. This example also shows a tunnel interface and a wireless adapter.

```

$ find_io
Interface Name eth1 -> MAC addr 0:6:29:15:49:f3 Flags=0
Interface Name eth0 -> MAC addr 0:16:41:50:b:93 Flags=0
Interface Name tap0 -> MAC addr 0:1a:6b:3a:60:c1 Flags=0
Interface Name ath0 -> MAC addr 0:1a:3b:5a:67:c1 Flags=0
===== Start of Chpid Registry Information=====
Total Interfaces Found = 2
Chpid Num      Type   State   Slot Num   Port Num   Card       Interface
-----
f0             0     808     ff         0          wired     eth1
f1             0     808     ff         1          wired     eth0
a0             0     808     ff         3          TunTap    tap0
f8             0     808     ff         4          wireless  wlan0

```

The MAC addresses may be useful for positive identification of which adapter is associated with which eth $n$  address. The appropriate CHPID number is used in the devmap for the OSA devices.

A tunnel interface (denoted by tap0 in this example) is always assigned CHPID A0. The tunnel interface exists only after the z1090 is first started. A `find_io` command issued before the `awsstart` command may not display the tunnel CHPID. A wireless interface is normally assigned CHPID F8.

The IP address used during OSA Express emulation is set by the TCP/IP PROFILE parameter for z/OS, or the equivalent, when using a different System z operating system.

The awssosa device manager can emulate QDIO or non-QDIO operation. The mode is selected by the TYPE parameter in the devmap. Type OSD specifies QDIO operation and type OSE specifies non-QDIO operation. Non-QDIO operation is often noted as *LCS operation* or *3172 operation*, although these descriptions are not exactly correct. Non-QDIO operation can involve TCP/IP or SNA (or both), although SNA usage is not supported with the 1090.

## 1.8 Dual boot

There is no explicit 1090 support for dual boot systems; a dual boot system should be transparent to 1090 operation. There are many ways to produce a dual boot system. Providing a base system for the 1090, whether it uses a dual boot setup or not, is up to the user. The following discussion describes one method, based on an earlier SUSE® Linux release and Windows® XP.

We started with a ThinkPad containing Microsoft® Windows XP installed in a partition that used the whole disk. (The internal disk was nominally 100 GB.) We needed to shrink the Windows partition to make room for our Linux partitions. Our first step was to obtain the PowerQuest Partition Magic product (version 8.0). Our version was resident on two diskettes.

Partition Magic displayed a single hard disk partition with the following characteristics:

Partition	Type	Size MB	Used MB	Unused MB	Status	Pri/Log
*:WindowsXP	NTFS	95393.0	11627.8	83765.0	Active	Primary

We used the option **Operations** → **Resize/Move** as follows:

```

Freespace Before: 0.0 MB
New Size          : 95393.0 MB          <---change to 30000
Free Space After:
<Apply> <Yes>

```

This changed the Windows partition to 30 GB, which would still provide over 18 GB of free space within the partition. The operation took several minutes and we received error messages during the operation:

```

Error #1627 Upcase table incorrect, File 10 (128)
<OK> Continue
Error #1627 Upcase table incorrect, File 10 (128)
<OK> Continue

```

The errors did not appear to create a problem. When the operation finished, Partition Magic displayed the following:

Partition	Type	Size MB	Used MB	Unused MB	Status	Pri/Log
*:WindowsXP	NTFS	29,996.3	11625.8	18,370.5	Active	Primary
*:	unalloc	65,397.0	0.0	0.0	None	Primary

We then rebooted Windows (twice) to allow it to update tables and verify that it still worked. At this point we had 65 GB available for Linux, the 1090, and System z emulated volumes.

We installed openSUSE Linux, and we selected **Partitioning** and **Base Partition on this Proposal** during installation. We created a partition table as follows:

Device	Size	F	Type	Mount	.....
/dev/sda	93.1GB		HTS....		
/dev/sda1	29.3GB		HPFS/NTFS	/windows/C	
/dev/sda2	63.8GB		Extended		
/dev/sda5	2.0GB	F	Swap		
/dev/sda6	8.0GB	F	ext3	/	
/dev/sda7	53.8GB	F	ext3	/z	

Take care, of course, not to delete or format the Windows partition while creating your own Linux partitions. The three partitions we created (sda5, sda6, sda7) correspond to the partition descriptions in Volume 2.

We then continued the Linux installation in the usual manner. After that was completed, we rebooted the ThinkPad. During startup, grub displayed the following menu:

```

SUSE Linux                      (default selection)
Windows
Failsafe -- SUSE Linux

```

If no selection is made within 8 seconds, Linux was booted by default. You can change this (running under Linux) as follows:

```

# gedit /boot/grub/menu.lst
# Modified by YaST2.....
color white/blue black/light-gray
default 0
timeout 8
...
### Don't change this comment...
title SUSE Linux ...
    root ...
    kernel ...

```

```

        initrd ...
        ...
title Windows
        ...
title Failsafe -- SUSE Linux
        ...

```

Change the default number to 1 if you want to boot Windows by default. Change the timeout value (which is in seconds) to a larger number if you want more time to make the selection after booting the PC.

## 1.9 Remote operation

A 1090 (including z/OS) may be operated remotely, using a Linux command window (telnet or VNC or ssh) and TN3270e sessions connected to the base Linux on the 1090 system. No special techniques or setups are required. As with local operation, it is important to have the TN3270e session for the z/OS console connected before IPLing z/OS.

Security considerations in your environment determine whether simple telnet or the more secure ssh should be used for the Linux command windows used to control the 1090.

## 1.10 x3270 cursor position

The 3270 cursor in an x3270 cursor window may be positioned incorrectly if the window is *clicked* to activate it. We used two methods to partly bypass this inconvenience:

- ▶ Click the top title bar of the x3270 window. This activates it (and *raises* or redraws it, if needed) without moving the 3270 cursor in the window.
- ▶ On SUSE, go to **Computer** → **Control Center** → **Windows** and check the option “Select windows when the mouse moves over them.” This option is most helpful when the x3270 window is not partly overlaid by another window.
- ▶ On Red Hat®, go to **System** → **Preferences** → **Windows** and check the option to select a window when the mouse moves over it.

## 1.11 Devices, memory, msgmni, ulimit

The current version of the 1090 has a maximum of 1024 emulated I/O devices. There is an interaction between the actual number of I/O devices and the amount of memory available for System z usage.<sup>10</sup> Each defined I/O device requires approximately 300 KB of shared virtual memory in the Linux address spaces. In most cases, where the number of defined I/O devices is small<sup>11</sup>, this is not a significant memory component. However, when the number of emulated I/O devices grows into the hundreds, the amount of shared memory consumed becomes significant. In such cases it might not be possible to allocate the maximum System z memory (2047 KB for a 32-bit version of the 1090). This should not be a problem with a 64-bit system, but it can become a significant configuration factor in a 32-bit system.

There is also an interaction between the number of emulated I/O devices and the kernel.msgmni value. Our installation instructions set this value to 512, which is suitable for

<sup>10</sup> This discussion is primarily for 32-bit Linux systems.

<sup>11</sup> By “small” we mean numbers in the general range of 25 to 50 I/O devices. For example, a devmap with 40 devices will cause about 10 MB of shared virtual memory to be used for I/O device support.

smaller systems. If you have more than, say, 64 emulated I/O devices defined in a devmap, then use the following formula:

```
kernel.msgmni = (350 + 3 * number-of-I/O-devices)
```

This is not an exact formula, but it should produce safe values. Also, if you have more than approximately 128 emulated I/O devices you should use `ulimit -m` and `-v` statements mentioned in Volume 2.

## 1.12 Linux ftp and telnet

You may want to activate ftp and telnet on your Linux system, assuming you included them during installation.<sup>12</sup> The openSUSE path for telnet (if telnet is available)<sup>13</sup> is **Control Center** → **YaST** → **Network Services** → **Network Service (xinetd)** → **select telnet** → **select Toggle On**. The path for vsftpd is almost the same, provided you installed vsftpd and it is not in listen mode.

After enabling ftp, you need to edit `/etc/vsftpd.conf` and make the following changes:

```
write_enable = YES
ftp_banner = "1090 Linux Base"           (optional text)
local_enable = YES                       (optional)
LISTEN = NO                              (if you use xinetd to start it)
```

You may need to customize or disable the firewall in order to use these services. This starts at **Control Center** → **YaST** → **Security and Users** → **Firewall**. A firewall is a good thing, of course, when operating on a public network. If you are on an isolated network, you might simply select **Service Start: Manually** and **Stop Firewall Now**.

For Red Hat, similar enablement is available via **System** → **Administration** → **Server Settings** → **Services** → **On Demand Services**. Our most recent RHEL 5.3 installation did not have ftp installed.

## 1.13 Startup scripts

We created several trivial startup scripts for our z/OS 1090 system, such as the following:

```
$ gedit start00
  cd /home/ibmsys1
  awsstart aprof1
  sleep 4
  echo 1090 started
  x3270 -port 3270 mstcon@localhost &
  sleep 2
  x3270 -port 3270 tso@localhost &
  sleep 2
  ip1 a80 parm 0a8200
  sleep 2
  echo IPL issued
```

<sup>12</sup> This description is for SUSE 10.1 and may change for later releases. Likewise, the Red Hat description is for RHEL 4.4 and may change in later releases.

<sup>13</sup> Recent Linux distributions appear to have dropped telnet.

We could then start our system with a single `./start00` command. These scripts (differing only in the IPL parm data) were trivial and could be enhanced in many ways.





## Tapes (SCSI and awstape)

Tape drive usage (for real SCSI tape drives and for emulated tape drives using awstape formats) is important to many developers. The 1090 offers a number of options in this area.

## 2.1 SCSI tape drives

In general, zPDT supports the use of SCSI tape drives. However, not all SCSI tape drives may be usable. The usability depends on the exact tape drive model, the firmware level, the firmware options selected, and the exact SCSI adapter (and firmware level) that is used. IBM has tested a number of tape drives, but cannot guarantee that *your* tape drive will work. We strongly suggest that you work with your zPDT supplier to understand your SCSI tape drive circumstances.

SCSI tape drives may be used in two ways:

- ▶ The `awsscsi` device manager allows SCSI tape drives to be used by System z programs.
- ▶ Several Linux utilities that directly use SCSI tape drives are provided with the 1090. These utilities are normally used when the 1090 function is not active. These utilities are not associated with devmaps or a device manager.

### 2.1.1 The `awsscsi` device manager

Selected SCSI tape drives may be used as “real” tape drives during 1090 operation. The `awsscsi` device manager is used and allows the SCSI tape drive to appear as a 3420, 3480, or 3490 device. A typical devmap definition might be:

```
[manager]
name awsscsi 7000
device 581 3490 3490 /dev/sg5
```

The 7000 in this example is the arbitrary CUNUMBR operand. This example defines a tape drive at address (device number) 581. If z/OS is being used, then the current z/OS IODF must have a corresponding device (3490) defined for this address. (z/VM detects devices dynamically and would find a 3490 at this address.)

The Linux device for the SCSI tape drive can be changed with the `awsmount` command. For example,

```
$ awsmount 580 -u /dev/sg5          (disassociate sg5)
$ awsmount 580 -m /dev/sg3          (mount different drive)
```

The last operand of the device statement (`/dev/sg5` in the example) denotes the SCSI device to be used. Determining this operand is a bit complicated. Linux can address a SCSI tape drive in three ways:

```
/dev/stN          (where N starts at 0 and is incremented as needed)
/dev/nstN
/dev/sgN
```

The `/dev/stN` and `/dev/nstN` interfaces are for *sequential tape devices*. The first tape drive on a Linux system would be `/dev/st0`; a second tape drive would be `/dev/st1`, and so forth. The two forms, `/dev/stN` and `/dev/nstN`, differ only in whether a rewind is performed when the device is closed.<sup>1</sup> The `/dev/stN` and `/dev/nstN` interfaces are used with the 1090 stand-alone tape utility functions, such as `scsi2tape`, `tape2scsi`, and also by Linux utilities such as `tar` and `mt`. The `/dev/stN` and `/dev/nstN` interfaces are *not* used with the `awsscsi` device manager.

The `/dev/sgN` devices are *general SCSI devices* and the `awsscsi` device manager uses the `/dev/sgN` interfaces.<sup>2</sup> Unfortunately the N value for a given device is typically not the same in

<sup>1</sup> The `/dev/stN` device automatically rewinds (whatever this may mean for the actual device) when the device is closed. The `/dev/nstN` devices do not provide an automatic rewind.

the stN and the sgN forms. All Linux SCSI devices are assigned sgN numbers, and Linux treats many of its normal devices as SCSI (even if they are not really hardware SCSI devices).<sup>3</sup> The first (and only) tape drive on a Linux system would be /dev/st0 but it might be /dev/sg7, for example.

The easiest way to determine what the sg device number represents is to list /proc/scsi/scsi. For example:

```
$ cat /proc/scsi/scsi
Attached devices:
Host: scsi0 Channel: 00 Id: 00 Lun: 00      (this is sg0)
  Vendor: IBM      Model: root      Rev: V1.0
  Type:   Direct-Access      ANSI SCSI revision: 02
Host: scsi2 Channel: 00 Id: 00 Lun: 00      (this is sg1)
  Vendor: IBM      Model: 03592E05    Rev: 1C91
  Type:   Sequential-Access  ANSI SCSI revision: 03
```

This example shows two SCSI devices (a disk and a tape) which would correspond to /dev/sg0 and /dev/sg1. In this case, /dev/sg1 is the device you would specify for the awsscsi device manager, but any Linux utility would use /dev/st0 or /dev/nst0 for the same tape drive. The devices listed in /proc/scsi/scsi are in sgN order; sg0 is first, sg1 is second, sg2 is third, and so forth.

There is another issue with both /dev/stN and /dev/sgN devices: the permission bits must allow usage by the 1090. Some Linux systems allow general user access to /dev/stN devices by default; no Linux systems allow general access to /dev/sgN devices. You must change the permissions to allow general access to your device. For example:

```
$ ls -al /dev/sg*
crw-r----- 1 root disk 21, 0 2008-09-03 14:44 /dev/sg0
crw-rw---- 1 root tape 23, 0,2008-99-03 14:44 /dev/sg1
$ su
(enter root password)
# chmod 666 /dev/sg1
# exit
$ ls -al /dev/sg*
crw-r----- 1 root disk 21, 0 2008-09-03 14:44 /dev/sg0
crw-rw-rw-+1 root tape 23, 0,2008-99-03 14:44 /dev/sg1
```

In this example, we changed the permissions for /dev/sg1 so that everyone (including the 1090) can read and write to it. This change (by manually entering **chmod**) is lost when Linux is rebooted, but is suitable for many situations. Always verify the correct sgN number first, by listing /proc/scsi/scsi.

A more permanent change can be made by modifying Linux boot functions. Unfortunately, there are two problems with this:

- ▶ The method of modifying Linux boot functions differs with different Linux distributions. For example, we might change /etc/rc.local, or /etc/rc.d/rc.local, or /etc/init.d/boot.local, or some other file, depending on the exact Linux distribution.
- ▶ We can easily make a general change for stN and nstN devices, but we cannot easily make a general change for sgN devices. We should not change the permissions to allow

<sup>2</sup> The reason is that the /dev/stN interface does not accept the full SCSI command set that is needed for some tape operations. The /dev/sgN interface allows all SCSI tape commands to be passed to the tape drive.

<sup>3</sup> The sg numbers are assigned in ascending order by bus and by device on the bus. However, since Linux treats a number of non-SCSI devices as though they were SCSI, these bus and device numbers are difficult to predict in advance.

universal access to all /dev/sgN devices; this would allow easy destruction of our Linux system.

We could place the following lines in /etc/init.d/boot.local (assuming our particular Linux uses this file):

```
chmod 666 /dev/st[0-9]           change all tape devices
chmod 666 /dev/nst[0-9]         change all tape devices
chmod 666 /dev/sg7              change particular SCSI device
```

Do not use **chmod 666 /dev/sg[0-9]** because this would allow anyone to directly access all the SCSI devices in your system. Unfortunately, the sgN numbers are dynamically assigned during Linux boot processing and depend on what devices are found (powered up) during boot. Unless you always have exactly the same devices powered up when you boot Linux, you cannot safely predict the sgN number of a given device.

## Drives and interfaces

The only SCSI tape drives used during development were the Fujitsu 3490E drive, the IBM 3580 LTO drives, and the IBM TS1120 drives. Although such usage is untested by IBM, other SCSI tape drives, such as DLT units, *may* operate correctly.

A key consideration for traditional (“parallel” interface) SCSI tape drives is the hardware interface. These interfaces include HVD (high voltage differential), LVD (low voltage differential), and single-ended (narrow or wide). The tape drive interface must match the SCSI adapter interface in your PC, and the underlying Linux must work correctly with the tape drive.

IBM used the following two Fibre SCSI adapter cards for IBM TS1120 operation:

- ▶ Fibre Channel: Emulex Corporation Zephyr-X LightPulse Fibre Channel Host Adapter (rev 02)
- ▶ Fibre Channel: QLogic Corp. ISP2432-based 4Gb Fibre Channel to PCI Express HBA (rev 03)

In both cases the Linux drivers defaulted to a maximum block size of 32K. Block sizes larger than 32 K are typically not used by application programs; however, large block sizes may be used by backup programs (such as ADRDSSU for z/OS) or by virtual tape managers.

The following Linux commands were used to set def\_reserved\_size to 65536:

```
$ su                               (change to root)
# rmod sg                           (removes the sg module)
# /sbin/modprobe sg def_reserved_size=65536
    (loads the sg module with the default reserved size up to 64k)
# cat /proc/scsi/sg/def_reserved_size
    (displays the current setting for def_reserved_size)
# exit                               (leave root)
```

This change allowed both cards to use 64k reserved buffer size for data transfers.

Note that these commands do not make a permanent change. A Linux reboot puts the default size back to 32768.

## 2.2 SCSI utilities

Two 1090 utilities can work directly with SCSI tape drives,<sup>4</sup> assuming the Linux system has access to the SCSI adapter. Awstape files (in Linux) can be moved to and from SCSI tape devices using the **scsi2tape** and **tape2scsi** commands.

```
$ scsi2tape /dev/st0 /z/my/TAPE23      copy SCSI tape to awstape file
$ tape2scsi /my/tapes/111111 /dev/st0  write SCSI tape from awstape file
$ scsi2tape -c /dev/st0 /z/mytape2     compress the awstape file
```

The two commands mentioned here are typically used when the 1090 is not active.<sup>5</sup> The awstape files used by these commands can then be used with emulated tape drives.

## 2.3 awstape utilities

1090 users often build a substantial “tape” library on disk, all in awstape format. The **tapeCheck** command may be used to verify that a file (which corresponds to a tape volume) is in the correct awstape format.

```
$ tapeCheck /z/TAPE01                verify format of awstape file
```

The **tape2file** command copies an awstape file to a simple byte stream in a Linux file, removing the awstape control blocks within the file.

```
$ tape2file /z/mytape /tmp/filex
```

The **card2tape** command copies a Linux text file (in ASCII or EBCDIC) to an awstape file as 80-byte records, using the same conversion conventions as the AWSRDR device manager:

```
$ card2tape /tmp/myLinux.stuff /z/tape01      copy without translation
$ card2tape -a /tmp/myLinux.xyz /z/tape01    force translate ASCII to EBCDIC
```

The **tape2tape** command copies an emulated tape volume (awstape file in Linux) to another emulated tape volume (another awstape file in Linux):

```
$ tape2tape /tmp/old.tape /z/new.tape
$ tape2tape -i -s /tmp/old.tape             scan and summarize tape content
$ tape2tape -c /tmp/old.tape /mine/new.tape copy and compress
```

The **-s** flag (scan flag) prevents creation of an output file. The **-i** flag displays a summary of the contents of the input tape. This command is normally used to compress or uncompress an awstape volume, or to scan the content. A simple copy of an emulated tape volume (without any additional processing) is easily done with the Linux **cp** command.

The **tapePrint** command lists the contents of an emulated tape volume. Data is displayed in hex and character format. The characters are assumed to be EBCDIC unless the **-a** flag is used.

```
$ tapePrint /tmp/my.awstape.file
$ tapePrint -a /tmp/my.awstape.file | more
```

Both the **card2tape** and **tape2tape** commands can produce compressed awstape files.

<sup>4</sup> In principle, any SCSI tape device that can be used by Linux may be used, although only a small set of drive types are officially supported by the 1090.

<sup>5</sup> If the SCSI devices are not in the active devmap, these utilities can be safely used while zPDT is active.





## **z/OS notes**

This chapter is not intended as a general z/OS guide, or as a guide to the AD-CD systems. However, a few common questions or problems are discussed here.

## 3.1 z/OS CP and memory display

The `d m=cpu` command should display the following information:

```
d m=cpu
ID CPU SERIAL
00 + 000971090
CPC SI = 1090.306.IBM.02.000000000000097

d m=stor
REAL STORAGE STATUS
ONLINE-NOT RECONFIGURABLE
0M-1500M
```

where 1090 is the System z machine type and 97 is the System z serial number assigned by our 1090 USB hardware key. Each 1090 hardware key assigns a different System z serial number.

## 3.2 z/OS spin loop timeouts

A z/OS spinloop may time out and produce an S071 ABEND. This is because the default timeout is set for a faster “real” processor. You can change this time value by creating or altering member EXSPATxx in PARMLIB:

```
member EXSPAT00
SPINRCVY ABEND
SPINTIME=60
```

In the z/OS 1.10 AD system we created ADCD.Z110.PARMLIB member EXSPAT00 and then issued the MVS console command `SET EXS=00` to immediately activate the member. It will be picked up automatically at IPL time. The 60 second value shown here is arbitrary, but should be safe. We suggest this not be done unless you experience spinloop problems.

## 3.3 Larger panel

The x3270 terminal emulator can be started with an optional parameter, as follows:

```
$ x3270 -port 3270 -oversize 133x60 localhost &
```

This produces a 3270 window with 133 columns and 60 lines. (Other sizes may be specified; this is simply an example.) Basic TSO does not use the “extra” window space. ISPF can use it if max is specified for the window format in the ISPF option 0 panel. (ISPF does not use the extra width unless a data set being displayed or edited has records that can use the extra width.)

## 3.4 z/OS disk space

Some z/OS functions assume certain disk allocations. For example, with the z/OS 1.10 AD system the RRS function looks for logger room on volume ZASYS1. z/OS also places SVC dumps on this volume and a number of these dumps can exhaust the free space on the

volume. You can look for data sets with names such as SYS1.ADCC.DMPnnnnn and delete them (assuming you are not working on a problem that involves these dumps).

The exact data set name pattern for SVC dumps is sent in the appropriate COMMNDxx member in PARMLIB.

## 3.5 Java and WebSphere Application Server startup

The IBM Java Virtual Machine (JVM) for zSeries is designed and optimized for best throughput performance on large mainframes. zPDT is sufficiently different in capabilities and characteristics from traditional System z hardware so that certain Java applications may not perform at their best without some tweaking of the JVM. This has been found especially true for startup and installation types of workloads. Startup time improvements of 30-40% have been seen on certain applications.<sup>1</sup>

Be aware that, with the z/OS AD-CD system, some IPL options include Java in their configuration and others do not. If you cannot find `java` on your system, you may not have used an IPL option that loads it.

This chapter provides some tips for improving the performance of Java-based applications. They may or may not be useful in your particular Java use-case. A specific example is given for tweaking the startup performance of IBM WebSphere Application Server v6.1.

### About Java versions

The exact version of Java that your application uses greatly affects how you can go about tweaking it. If you have Java installed, you can check the version on your system by running the `java` executable from an OMVS prompt:

```
$ (cd to the directory with the Java bin files2)
$ (if Java is in your search path there is no need to cd to it)
$ java -version
Java(TM) SE Runtime Environment (build pxz6460sr6-20090729_05(SR6))
IBM J9 VM (build 2.6, JRE 1.6.0 IBM J9 2.6 Linux s390x-64 20090731_039920
(JIT enabled, AOT enabled)
J9VM - 20090731_039920
JIT - dev_20090804_1730
GC - R26_head_20090731_1122_B39896
J9CL - 20090715_1250)
JCL - 20090727_01
```

The tips presented in this section apply specifically to the current IBM J9 Java Virtual Machine. The older *Classic* or *Sovereign* JVMs are not relevant.

Unless you are using a JVM bundled with a middleware application, you should try to install and use the latest version of the IBM JVM, because it is likely to give you the best performance.

Also, note that many Java-based middleware applications ship with their own copies of the JVM. As a first step you will have to determine exactly which JVM is being invoked by your application. Furthermore, you may have to locate either the startup script or the JVM options

<sup>1</sup> Mitch Johnson (IBM USA) has investigated ways to improve IBM WebSphere Application Server startup times on smaller System z machines, including the 1090. This section is based on his work.

<sup>2</sup> In our system we found a set of Java executables in `/usr/lpp/java/J6.0/bin`. This was after IPLing our AD system with the "BC" parameter.

file that the middleware may be using. Unfortunately, this varies from application to application and cannot be generally documented.

## General principles

The JVM used with z/OS is tuned for large systems and maximum throughput, rather than being tuned for a quick startup. It does not perform too well on zPDT as far as startup performance is concerned; startup of a GUI-based application may seem quite sluggish. The primary way to improve this is by telling the JVM to optimize for startup performance rather than throughput. In a nutshell, the most important change is to provide the '-Xquickstart' option to the JVM. This can be done in two ways.

### JVM command line

If you can locate the command line being used by your program to invoke the JVM, you can simply add this option near the start of the JVM command statement. This command line may be present in the start-up script of the application. For example,

```
$ java -Xquickstart <other JVM options> MyProgram <program options>
```

This is the best method to provide the option. However, it may not be trivial to locate the exact startup script that launches your application and the location of the JVM command inside it.

### Environment variable

If you cannot locate the JVM command line responsible for running your application, you can achieve a similar effect by setting an environment variable. Note that this environment variable must be set inside the OS (zOS, Linux for System z) running inside zPDT.

Using the shell, you can run the following command before launching your application

```
$ export IBM_JAVA_OPTIONS="-Xquickstart"
```

Or you can add this option to the default shell profile on a system-wide basis by adding the following line to /etc/profile:

```
export IBM_JAVA_OPTIONS="-Xquickstart"
```

While this is a good setting to set system-wide, we suggest that you use this mechanism in addition to customizing the startup script or JVM command line for your application. Certain Java startup jobs may not inherit the bash profile environment variables while others may override the value being specified in the IBM\_JAVA\_OPTIONS.

## IBM WebSphere Application Server V6.1 on ADCD 1.10

IBM WebSphere Application Server 6.1 ships with its own JVM in the /usr/lpp/zWebSphere/V6R1M0/java directory. The exact path may be different based on the version of WebSphere Application Server you are using.

There are several aspects of WebSphere Application Server that need to be customized.

### wsadmin tool

The **wsadmin** tool is used to configure and administer application servers, application deployment, and server run-time operations.

To improve **wsadmin** performance on zPDT, edit the wsadmin.sh script located in the /usr/lpp/zWebSphere/V6R1M0/bin directory by adding the *-Xquickstart* option for the z/OS platform. (If you use the jython script described later you do not need to make the individual option and configuration changes noted here.)

```
case $PLATFORM in
```

```

AIX)
    PERF_JVM_OPTIONS="-Xms256m -Xmx256m -Xquickstart" ;;
Linux)
    PERF_JVM_OPTIONS="-Xms256m -Xmx256m -Xj9 -Xquickstart" ;;
SunOS)
    PERF_JVM_OPTIONS="-Xms256m -Xmx256m -XX:PermSize=40m" ;;
HP-UX)
    PERF_JVM_OPTIONS="-Xms256m -Xmx256m -XX:PermSize=40m" ;;
OS/390)
    PERF_JVM_OPTIONS="-Xms256m -Xmx256m -Xquickstart" ;;
esac

```

### **WebSphere address spaces**

The WebSphere Application Server runs in multiple address spaces with a separate JVM in each address space. There is one control region, zero or one adjunct region, and at least one servant region. The Java properties of these address spaces can be tweaked by editing files `control.jvm.options`, `adjunct.jvm.options` and `servant.jvm.options` in the `/u/WebSphere/V6R1/ADCD.ADCD.BBOS001` and adding the `"-Xquickstart"` option:

```

...
-Dosgi.configuration.area=/u/WebSphere/V6R1/AppServer/profiles/default/configuration
-Xquickstart
-Dserver.root=/u/WebSphere/V6R1/AppServer/profiles/default
...

```

Note that if the *admin console* is used to modify and save the configuration, the changes to the properties files will be overwritten.

### **The jython script**

You can make these changes permanent by creating the following script, *in ASCII*, in `/u/ibmuser/jvmprop.py`<sup>3</sup>:

```

#####
# Script to set JVM generic arguments to
#           -Xjit:optLevel=cold,disableInterpreterProfiling
# to improve startup performance of WebSphere on zPDT images.
#
# Author: Mitch Johnson
# Organization: IBM lab services for WebSphere
#####

import java.lang.System as sys
lineSeparator = sys.getProperty('line.separator')
global AdminApp, AdminConfig

cell=AdminConfig.list("Cell")

node=AdminConfig.list("Node",cell)

serverList=AdminConfig.list("Server",node).split(lineSeparator)

for server in serverList:
    serverName=AdminConfig.showAttribute(server,"name")

```

<sup>3</sup> Another convenient directory could be used; the location of the script is not critical.

```

print "Updating genericJVMArguments for server: " + serverName

objID=AdminConfig.getid("/Server:" + serverName + "/")

jvmList=AdminConfig.list("JavaVirtualMachine",objID).split(lineSeparator)

for jvm in jvmList:
    print "Updating JVM generic arguments for " + jvm
    AdminConfig.modify(jvm, [{"genericJvmArguments","-Xquickstart
-Xverify:none -Xjit:optLevel=cold,disableInterpreterProfiling"}] )

print AdminConfig.queryChanges()
print AdminConfig.save()
print "Changes saved"

```

Remember: the above script must be entered in ASCII.

During the configuration of an instance of WebSphere on z/OS the system programmer specifies an HFS directory where the configuration files reside. We use /u/WebSphere/V6R1 in this example. The WebSphere configuration process populates this structure with the customized configuration and symbolic links to the WebSphere product code in /usr/lpp/zWebSphere. The jython script shown above does not connect to an active WebSphere instance; instead, it makes changes to the configuration files relative to the directory in which its execution begins. Therefore, our example assumes that the configuration files are located in directory /u/WebSphere/V6R1.

Go to directory /u/WebSphere/V6R1/AppServer/bin and enter the following command:

```
./wsadmin.sh -conntype none -f /u/ibmuser/jvmprop.py -lan jython
```

This command may take several seconds to execute. It updates each server's *genericJVMArgument* value.

## References

The following two URLs may be used to obtain more JVM information related to startup performance and diagnostic techniques:

- ▶ IBM Java 5 Diagnostic Guide  
<http://publib.boulder.ibm.com/infocenter/javasdk/v5r0/index.jsp>
- ▶ CICS and JVM Startup Time  
<http://publib.boulder.ibm.com/infocenter/cicsts/v2r2/index.jsp%3Ftopic%3D/com.ibm.cics.ts22.doc/dfhpj/dfhpj93.htm>

## 3.6 Stand-alone z/OS dump

The z/OS “stand-alone dump” (SAD) is a program that is IPLed from tape or disk. It does not run under z/OS. However, it assumes that z/OS is present in System z memory at the time the stand-alone dump is IPLed. The stand-alone dump program is sensitive to the release of z/OS that is being used and should match the z/OS release. That is, a new version of the stand-alone dump program should be generated whenever a new release of z/OS is installed.

This section describes a simplified use of stand-alone dump based on the AD z/OS system release 1.9. Two (emulated) tape drives are used. One is the residence volume of the stand-alone dump program and the other is the output device for the dump.

The example here uses a tape for dump output. We assume you are running under zPDT and that an emulated tape drive will be used. This results in a dump in an awstape file and this can be exported from Linux in a variety of ways. Our specific examples are for the AD-CD z/OS 1.9 system; other z/OS systems may require JCL changes.

## Devmap

The devmap should have tape devices defined. Only one device (580 in the example) is needed when generating the IPLable stand-alone dump program. Two devices are needed when using the program. The devmap might have the following definitions:

```
[manager]
name awstape 1234
device 580 3490 3490
device 5FF 3490 3490
```

The job used to generate the stand-alone dump program runs under z/OS and the output device must be known to z/OS. The z/OS 1.9 AD system has a 3490 tape drive at address 580, and this is used when generating the dump program.

It is not necessary to use an address known to z/OS when IPLing the dump program, since the dump program does not run under z/OS. However, our example also uses address 580 to IPL and run the dump program. Address 5FF is arbitrary; it is unknown to our z/OS, although it would not matter if it were known to z/OS.

### 3.6.1 Generating a stand-alone dump program

The following job generates the stand-alone dump program and writes it on tape. You later IPL it from this tape when you want to use it.

```
//SADBILL JOB 1,OGDEN,MSGCLASS=X
// EXEC PGM=AMDSAOSG
//SYSLIB DD SYS1.MACLIB,DISP=SHR
// DD SYS1.MODGEN,DISP=SHR
//IPLDEV DD UNIT=580,DISP=(NEW,KEEP),LABEL=(1,NL),
// DCB=(BLKSIZE=12288,RECFM=U,DSORG=PS)
//GENPRINT DD SYSOUT=*
//GENPARMS DD *
        AMDSADMP IPL=T580,OUTPUT=T5FF,VOLSER=SADMPR,CONSOLE=(0700,3279),      X
        COMPACT=NO
/*
```

This job will call for a tape mount when it runs. We arbitrarily placed the awstape volume in the /z directory and named it SADTAPE.

```
$ awsmount 580 -m /z/SADTAPE
```

Check the JES2 output when the job ends to ensure it completed correctly.

### 3.6.2 Taking a stand-alone z/OS dump

Your devmap must contain two 3490 tape drives, as explained earlier, in addition to whatever devices are used by z/OS. Also, the dump program we generated assumes that a 3270 console at address 700 will be used.

We assume you have been running z/OS and need to take a stand-alone dump for some reason:

1. Mount the stand-alone dump residence tape and the output tape.

```
$ awsmount 580 -m /z/SADTAPE
$ awsmount 5FF -m /tmp/SADOUT
```

2. Issue **stop** and **storestatus** commands for System z. (This is needed to capture the contents of various registers).

```
$ stop
$ storestatus
```

3. IPL the stand-alone dump program from tape.

```
$ ip1 580
```

4. Wait about 10 seconds and press Enter on the 3270 console at address 700.

```
AMD083I AMDSADMP: STAND-ALONE DUMP INITIALIZED. IPLDEV: 0580 LOADP:
AMD001A SPECIFY OUTPUT DEVICE ADDRESS (1): 5ff
AMD101I OUTPUT DEVICE: 05FF
      SENSE ID DATA: FF 3490 10 3490 40  BLOCKSIZE: 29,120
AMD011A TITLE=my dump stuff
AMD005I DUMPING OF READ STORAGE NOW IN PROGRESS
AMD005I DUMPING OF PAGE FRAME TABLE COMPLETED
      etc
AMD029D REPLY W TO WAIT AFTER NEXT FULL SCREEN, ELSE REPLY N; REPLY=w
      etc
```

The file name used for the emulated output tape (/tmp/SADOUT) is completely arbitrary. However, remember that the dump output can be rather large; be certain there is sufficient Linux disk space to hold the output file. The message:

```
AMD033I I/O ERROR ON 5FF CMD=01 STATUS=0E00
      COND=UNKNOWN
```

probably means that you ran out of Linux disk space for the awstape output file.

## 3.7 Moving 3390 volumes

The following text describes a generic method of moving 3390 volumes between z/OS systems (including z/OS on zPDT). Another method, using a client/server application provided with zPDT, is described in “DASD volume migration” on page 123.

zPDT-emulated DASD volumes can be transferred to other systems in several ways. Sending a volume to another zPDT system is especially easy. The Linux file that holds the emulated 3390 volume can simply be copied. Optionally, the copy could be compressed (with **gzip**, for example) for transmission. The transmission could be by ftp, by a USB thumb drive, by burning a CD or DVD, or by various other means. The key element is that a large Linux binary file is being transferred.

Moving an emulated 3390 volume to (or from) a non-zPDT system is a little more complex, because it must be handled in a System z format instead of a Linux format. The traditional method is to dump the volume to tape (using the ADRDSSU program) and then restore the tape on the target system. This method can also be used with emulated tapes (in awstape format), provided that both the sending and receiving systems can use this format.

This section describes another method for moving a volume from z/OS on a zPDT system to a non-zPDT system. This example assumes the target system cannot use awstape format (otherwise we would use the easier method of creating a dump tape in awstape format). We assume there is a network connection between the zPDT Linux base system and the target z/OS system.

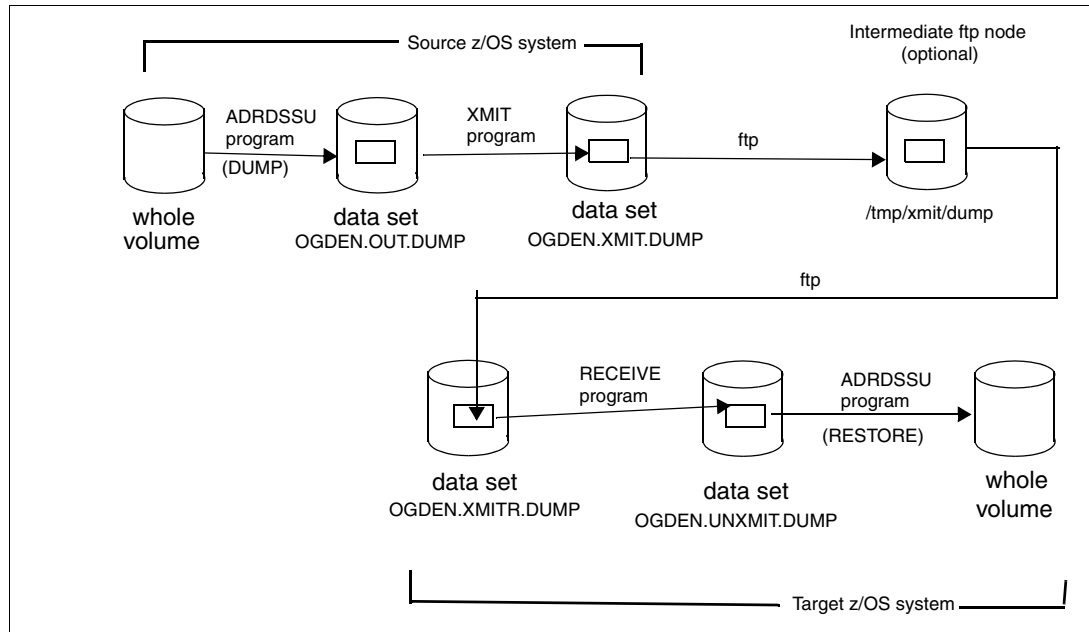


Figure 3-1 Overview of our example

## Preparation

We need z/OS disk space to hold a 3390 volume dump and to hold a reformatted volume dump. These can be large data sets. You may have sufficient space on existing z/OS volumes; we elected to create three new volumes for holding large temporary data sets. We did this using normal zPDT techniques. First we created three new emulated 3390 volumes (done while zPDT is not operational). The placement (/z), model (3390-3), and names of the files (TEMPnn) are all arbitrary.

```
$ alckkd /z/TEMP01 -d3390-3
$ alckkd /z/TEMP02 -d3390-3
$ alckkd /z/TEMP03 -d3390-3
```

We then added these volumes to our devmap. The addresses specified (AA0, AA1, and AA2) are unused address that are known as 3390 devices for our z/OS. (That is, z/OS has these addresses specified as 3390 devices in the IODF it uses during IPL. The AAx addresses are suitable for the default IODF in the z/OS AD systems.)

```
[manager]
name awsckd 0001
....
....
device AA0 3390 3990 /z/TEMP01
device AA1 3390 3990 /z/TEMP02
device AA2 3390 3990 /z/TEMP03
```

We then started zPDT and IPLed z/OS. During z/OS startup the new devices are recognized as uninitialized volumes and are varied offline. When z/OS was ready, we ran a job to initialize the volumes:

```
//BILL123 JOB 1,OGDEN,MSGCLASS=X
// EXEC PGM=ICKDSF,REGION=40M
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
  INIT UNIT(AA0) NOVALIDATE NVFY VOLID(TEMP01) PURGE -
    VTOC(0,1,05)
  INIT UNIT(AA1) NOVALIDATE NVFY VOLID(TEMP02) PURGE -
    VTOC(0,1,05)
  INIT UNIT(AA2) NOVALIDATE NVFY VOLID(TEMP03) PURGE -
    VTOC(0,1,05)
/*
```

The z/OS operator must reply **u** to a ICK003D message for each volume. The volsers (TEMP01, and so forth) are the same as the Linux file names; this is not required but is a good practice. After the volumes are initialized, they can be varied online to z/OS, using the MVS console:

```
vary aa0-aa2,onl ine
```

### 3.7.1 Create source dump

A normal ADRDSSU job is used to dump the source volume. Our example uses WAS001 as the volser of the source volume:

```
//BILL456 JOB 1,OGDEN,MSGCLASS=X
// PGM=ADRDSSU,REGION=40M
//SYSPRINT DD SYSOUT=*
//IN DD UNIT=3390,VOL=SER=WAS001,DISP=SHR
//OUT DD UNIT=3390,VOL=SER=TEMP01,DISP=(NEW,CATLG),
//      DSN=OGDEN.OUT.DUMP,SPACE=(CYL,(200,200))
//SYSIN DD *
  DUMP INDD(IN) OUTDD(OUT) ADMINISTRATOR COMPRESS OPTIMIZE(4)
/*
```

The space specified in the output DD statement may need to be adjusted, depending on the contents of the source volume. We next created another data set with specific DCB attributes.<sup>4</sup> (This step could be done using ISPF 3.2 functions, but we used a batch job to provide better documentation.)

```
//BILL567 JOB 1,OGDEN,MSGCLASS=X
// PGM=IEFBR14
//MAKEIT DD UNIT=3390,VOL=SER=TEMP02,DISP=(NEW,CATLG),
//      DSN=OGDEN.XMIT.DUMP,SPACE=(CYL,(200,200)),
//      DCB=(LRECL=80,RECFM=FB,BLKSIZE=3120)
```

We then used the TSO **xmit** command to reformat the dump:

```
xmit x.y ds('ogden.out.dump') outdsn('ogden.xmit.dump')
```

This command can take considerable time if a full volume is being processed. The result of all this is a volume dump in a format known to z/OS, but also in a format (fixed block) that can be handled by ftp. The **x.y** positional operand is needed in **xmit**, but is meaningless in this example.

---

<sup>4</sup> These DCB attributes are used by XMIT.

## 3.7.2 Send dump to Linux

We then sent the xmit-formatted dump to Linux, using an ftp connection from Linux to z/OS. We used the zPDT tunnel facility for the connection in our example, but any TCP/IP connection to z/OS could be used. The IP address for z/OS is 10.1.1.2 in this example:

```
$ ftp 10.1.1.2
Name (10.1.1.2:ibmsys1): ibmuser
Password: xxxxxx
Remote system type is MVS
ftp> cd 'ogden'
ftp> lcd /tmp
ftp> bin
ftp> get 'xmit.dump'
ftp> bye
```

This example has the ftp connection initiated from the Linux side. It could be done from the z/OS side, provided the Linux system has an ftp server running.

The dump is now in /tmp/xmit.dump as a normal (large) Linux file. It can be transmitted elsewhere using any technique suitable for a large Linux file. It could be compressed (using **gzip**, for example.) At this point the dump (OGDEN.OUT.DUMP) and the reformatted dump (OGDEN.XMIT.DUMP) on the source z/OS system can be deleted if disk space is a concern.

You can skip this Linux step if there is a direct ftp connection between the source z/OS system and the target z/OS system.

## 3.7.3 Receive dump

There are fewer complications if two data sets are preallocated on the receiving z/OS system. One data set is the target of an ftp transfer from Linux (or some other source) and the other is for the output of the TSO RECEIVE function. This last data set is then the input to a RESTORE job.

```
//BILL678 JOB 1,OGDEN,MSGCLASS=X
// PGM=IEFBR14
//D1 DD UNIT=3390,VOL=SER=TEMP01,DISP=(NEW,CATLG),
// SPACE=(CYL,(200,200)),DSN=OGDEN.XMITR.DUMP,
// DCB=(LRECL=80,RECFM=FB,BLKSIZE=3120)
//D2 DD UNIT=3390,VOL=SER=TEMP02,DISP=(NEW,CATLG),
// SPACE=(CYL,(200,200)),DSN=OGDEN.UNXMIT.DUMP
```

The dump file can be sent from Linux to z/OS using ftp. (If the file was compressed (in Linux) it must be uncompressed before sending it to z/OS.) Our example uses IP address 10.1.1.2 for z/OS because, for demonstration purposes, we used the same zPDT z/OS system that we used to create the dump volume. In practice, this is likely to be a different z/OS system that might not be in a zPDT environment.

```
$ ftp 10.1.1.2
Name (10.1.1.2:ibmsys1): ibmuser
Password: xxxxxx
Remote system type is MVS
ftp> cd 'ogden'
ftp> lcd /tmp
ftp> bin
ftp> put xmit.dump xmitr.dump
ftp> bye
```

We then used TSO to reformat the dump into the original format created by ADRDSSU:

```
receive indsn('ogden.xmitr.dump')
(reply to the prompt with) DSN('ogden.unxmit.dump')
```

Finally, the volume can be restored in z/OS:

```
//BILL890 JOB 1,OGDEN,MSGCLASS=X
// PGM=ADRDSSU,REGION=40M
//SYSPRINT DD SYSOUT=*
//IN DD UNIT=3390,DSN=OGDEN.UNXMIT.DUMP,DISP=SHR
//OUT DD UNIT=3390,VOL=SER=TEMP03,DISP=OLD
//SYSIN DD *
    RESTORE INDDNAME(IN) OUTDDNAME(OUT) PURGE ADMINISTRATOR COPYVOLID
/*
```

You might not want the COPYVOLID parameter in this job, depending on your circumstances. You cannot have two disk volumes with the same volser online to z/OS at the same time. If you do not specify COPYVOLID then the existing volser (TEMP03 in the example) is retained. If you specify COPYVOLID and a volume with this volser is already online, the restored volume is taken offline after the restore operation is complete. (In our example, the restored volser would be WAS001.)

## Comments

Many variations are possible in this general process. For example, some of the z/OS preallocation of data sets can be skipped if your ftp supports **site** and **locsite** subcommands. While perhaps a bit longer than absolutely necessary, we think the process shown here should work in most situations. With minor changes it can be used in the other direction, to copy a volume from a non-zPDT machine to z/OS on zPDT.

This process can be used to port an older version of z/OS from a non-1090 system to a 1090.

### 3.7.4 Stand-alone restore

This section describes a method of porting a single z/OS volume from an external system than can create awstape volumes<sup>5</sup> to a 1090 environment that has no System z software installed. The single z/OS volume used in this example is the one-volume z/OS system that is distributed with the AD package. Although the process we describe here is not required for normal 1090 use, we describe it in considerable detail because portions may be useful in unusual circumstances.

There are several phases involved:

- ▶ Prepare stand-alone versions of ICKDSF and the ADRDSSU restore program on an existing z/OS system that can write emulated tape volumes in awstape format.
- ▶ Dump the selected z/OS volume to an awstape file, using the ADRDSSU program. Note that the first two phases can be run on much older OS/390® or z/OS versions, on a variety of machines.
- ▶ Use gzip (or a compatible program) to compress the dump file. This is needed if the transport medium is a CD because the dump file will not fit on a CD unless it is compressed. (If the transport method is FTP or DVD then this step may be skipped.)
- ▶ Burn a CD containing the two stand-alone System z programs (each in awstape format) and the gzipped dump file. Take this CD to the Linux 1090 system.

<sup>5</sup> This could be a system with native awstape capabilities, or a system with an informal utility program to convert a sequential file (the dump data set) into awstape format. Such informal utilities are not part of the 1090 package.

- ▶ Create the necessary 1090 devmap.
- ▶ Copy the three CD files to Linux and unzip the dump file.
- ▶ Use a 1090 utility to create an emulated 3390 volume.
- ▶ Start 1090 operation and a 3270 session, IPL the stand-alone ICKDSF program, and initialize the emulated 3390 volume.
- ▶ IPL the stand-alone restore program and restore the dump volume.
- ▶ IPL the one-volume z/OS to verify that the process worked.

## Build the CD

We created a CD on an external system that was capable of writing awstape files. This system had unit 560 defined as an emulated tape drive. Three z/OS jobs were involved, as follows:

```
//BILL1 JOB 1,OGDEN,MSGCLASS=X
//* CREATE STAND-ALONE ICKDSF
// EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY,DCB=BLKSIZE=80
//SYSUT1 DD DSN=SYS1.SAMPLIB(ICKSADSF),DISP=SHR
//SYSUT2 DD UNIT=560,LABEL=(1,NL),VOL=SER=SAINIT,DISP=(,KEEP),
//          DCB=(RECFM=F,LRECL=80,BLKSIZE=80)

//BILL2 JOB 1,OGDEN,MSGCLASS=X
//* CREATE STAND-ALONE RESTORE
// EXEC PGM=ADRDSSU,PARM='UTILMSG=YES'
//SYSPRINT DD SYSOUT=*
//SAMODS DD DSN=SYS1.SADRYLIB,DISP=SHR
//TAPEDD DD UNIT=560,LABEL=(1,NL),VOL=SER=SAREST,DISP=(,KEEP),
//          DCB=(DSORG=PS,RECFM=U,BLKSIZE=32760,LRECL=32760)
//SYSIN DD *
        BUILD SA INDD(SAMODS) OUTDD(TAPEDD) IPL(TAPE)
/*

//BILL3 JOB 1,OGDEN,MSGCLASS=X
// EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//IN DD UNIT=3390,VOL=SER=SARES1,DISP=SHR
//OUT DD UNIT=560,VOL=SER=DUMP01,DISP=(,KEEP),LABEL=(1,NL)
//SYSIN DD *
        DUMP INDD(IN) OUTDD(OUT) ALLDATA(*) ALLEXCP ADMINISTRATOR -
        COMPRESS OPTIMIZE(4)
/*
```

These jobs created three awstape files (emulated tape volumes) on the external system. We then compressed the large file containing the dump, using the command:

```
$ gzip -c /tapes/DUMP01 > DUMP01.gz
```

Finally, we used a standard program to burn the three files (containing the two small stand-alone programs and the compressed DUMP01 file) to a CD. We then took the CD to the 1090 machine. (A CD is not really required, of course. We could simply FTP the files to Linux on the 1090 machine if a suitable network is available.) The CD burning program we used created a CD title of "CDROM."

We also noted that the one-volume z/OS system recognizes a suitable 3390 address (device number) for the system residence as A8F, expects to find a z/OS console at address 700, and can use a local 3270 TSO terminal at address 701.

## Linux directories

We used the following arrangement on the 1090 machine:

- ▶ Our userid was *ibmsys1*; it was a member of group *ibmsys*.
- ▶ Our home directory was */home/ibmsys1*.
- ▶ Directory */z* was created to hold System z emulated disks and tapes. It is a separate file system (partition) that was created when Linux was installed and has ample free space.

## Read the CD

We copied the stand-alone programs from the CD and uncompressed the dump tape:

```
$ cd /z
$ ls /media/CDROM                                verify CD is readable
$ cp /media/CDROM/SAINIT /z/SAINIT                DASDI program
$ cp /media/CDROM/SAREST /z/SARESTOR            restore program
$ gunzip -c /media/CDROM/TAPE01 > /z/TAPE01      dump tape
$ ls -al                                          verify files are on disk
  SAINIT      1108042 bytes
  SARESTOR    232648 bytes
  TAPE01      956763522 bytes
(you may close the CD window and remove the CD)
```

## Create 1090 files

We created the new emulated disk volume on the 1090 system:

```
$ alckd /z/SARES1 -d3390-3                      resulting volume 2.8 GB
```

We created and checked */ibmsys1/aprofi.txt* for the devmap:

```
$ cd /home/ibmsys1
$ gedit aprofi.txt                               or use your favorite text editor
[system]
memory 400m
3270port 3270

[manager]
name aws3274 1
device 0700 3279 3274 L700
device 0701 3279 3274 L701

[manager]
name awstape 4
device 0580 3480 2803 /z/SAINIT
device 0581 3480 2803 /z/SARESTOR
device 0582 3480 2803 /z/TAPE01

[manager]
name awsckd 8
device 0a8F 3390 3990 /z/SARES1
$ awsckmap devmap.txt                          verify there are no errors
```

## Run stand-alone programs

We started 1090 operation:

```
$ cd /home/ibmsys1
$ awsstart aprofi.txt
```

We then created a 3270 session:

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

We IPLed the SAINIT program:

```
$ ipl 0580
(wait a few seconds and press Enter on the 3270 screen)
(message CLEAR SCREEN WHEN READY should appear)
(Alt-c performs the Clear Screen function for an unmodified x3270)
ICK005E DEFINE INPUT DEVICE, REPLY 'DDDD,CUU' or 'CONSOLE'
console
ICK005E DEFINE OUTPUT DEVICE, REPLY 'DDDD,CUU' or 'CONSOLE'
console
ENTER INPUT/COMMAND
init unit(0a8f) devtyp(3390) volid(SARES1) nvfy vtoc(1,1,14)
  (displays device information)
ENTER INPUT/COMMAND:
u
(clear screen, using Alt-c, as needed)
(wait for FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0)
```

We IPLed the SARESTOR program:

```
$ ipl 581
(wait a few seconds and press Enter on the 3270 screen)
(message CLEAR SCREEN WHEN READY should appear)
ADRY005E DEFINE INPUT DEVICE, REPLY 'DDDD,CUU' or 'CONSOLE'
console
ADRY005E DEFINE OUTPUT DEVICE, REPLY 'DDDD,CUU' or 'CONSOLE'
console
ENTER INPUT/COMMAND:
restore frmdv(tape) frmadr(0582) toadr(0a8f) noverify
...REPLY y TO ALTER VOLUME CONTENTS, ELSE N
y
(long pause for restore operation; several minutes. Disk light blinks.)
(clear screen when requested)
(ignore progress messages such as NEXT TRACK TO WRITE)
(wait for FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0)
```

## IPL z/OS

We started another 3270 session (for TSO):

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

And then we IPLed the one-volume z/OS system that we just restored:

```
$ ipl 0a8f parm 0a8fsa          use the indicated IPL parameter
```

## 3.8 IODF Changes with zPDT

For a larger System z, IODF and IOCDS creation are almost always done at the same time, working with HCD. This typically starts as follows:

- HCD (usually started from an ISPF panel)
1. Define, modify, or view configuration data
  3. Processors
  4. Control Units
  5. I/O Devices

The HCD functions verify that an allowable configuration is specified. That is, the processor (type and model), CHPIDs, and I/O devices must all be mutually allowable. This is a useful check for a larger System z, but it creates problems for zPDT.

A zPDT system does not use an IOCDS and does not understand many hardware CHPID details.<sup>6</sup> Furthermore, typical zPDT configurations are simply not compatible with the normal HCD verification and processing we would use with a larger System z. At the time of writing, I/O device configuration for a z/OS system on zPDT consists of a *software only* IODF generation, followed by the creation of a matching devmap. Many z/OS users are not immediately familiar with a *software only* IODF and we provide an overview of the topic here. Once started, it is considerably simpler than a “normal” IODF.

Assume we want to add 15 OSA devices starting at address 410 and an OSAD device at address 41F. Using the IODF99 that is provided with the z/OS 1.10 AD system, we would proceed as follows:

- HCD *(started via ISPF menu item M.4)*
1. Define, modify, or view configuration data  
I/O DEFINITION FILE 'SYS1.IODF99'
    1. Operating System Characteristics  
/ OS390 *(select configuration named 'OS390')*
      7. Work with attached devices  
*(This should produce a list of current devices)*
- F11** - Add

At this point you should have a panel that wants you to enter a new work IODF name. We entered the name SYS1.IODF77.WORK and volser ZASYS1 in this panel. The data set name should follow this pattern (although the 77 portion of the name is arbitrary) and the volser should be the volume that is specified in the IPL parameter (ZASYS1 in the z/OS 1.10 AD system).

This is followed by a panel to add devices to the new work IODF. We entered the following:

Specify or revise the following values.

```
Device number. . . . . 410    + (0000 - FFFF)
Number of devices . . . . 15
Device type. . . . . OSA    +
Serial number. . . . .
Description. . . . .
Volume serial number . . . _____ (for DASD)
Connected to CUs . . _____
```

Press Enter and again select (with a / character) the OS390 configuration. Select option 1 (to connect or change the new I/O devices). This is followed by a panel allowing you to alter

<sup>6</sup> It is possible to use an IOCDS with zPDT, but no documentation is provided for this usage.

default device parameters; you should take the default options unless you have a particular reason for changing them. This is followed by a panel to associate esoteric names with the new devices; you might use this for DASD or tape devices but probably not for any other types of devices. Press Enter and then select (with a / character) the OS390 configuration again.

Press F3 to return to the device list, and you should now see 410,15 in the list. Again select F11 (to add devices) and add a single OSAD device at address 41F, following the same steps used for the 15 OSA devices.

When your new I/O devices have been added to the list, use F3 three times to return to the initial HCD menu. You then need to process your new IODF file.

2. Activate or process configuration data
  - I/O DEFINITION FILE '**SYS1.IODF77.WORK**'
    1. Build production I/O definition file
      - (This may produce some warning messages, usually about esoteric tokens. Ignore these messages. **F3** to exit the warning panel.)

Command ==>

```
Date & Time . . . . . : 2008-07-14 13:14:51
User . . . . . : IBMUSER
I/O Definition file. . . : SYS1.IODF77.WORK
Change reference number.: 00018
***** *****TOP OF DATA *****
.....
***** *****BOTTOM OF DATA *****
```

Press F3 to exit this panel. The next panel allows you to name the new production IODF file:

```
Specify the following values, and chose how to continue.
Work IODF name . . . . . : 'SYS1.IODF77.WORK'
Production IODF name . . : 'SYS1.IODF77'
Continue using as current IODF:
1 1. The work IODF in use at present
   2. The new production IODF specified above (not valid for zPDT)
```

The next panel allows you to specify or revise these values; press Enter. This should produce the message PRODUCTION IODF SYS1.IODF77 CREATED. Use F3 several times to exit from HCD.

To use the new IODF you must alter one or more of the LOADxx members in SYS1.IPLPARM to refer to the new IODF. For example, edit member LOAD00 in SYS1.IPLPARM:

```
IODF      99 SYS1                <--change this line
SYSCAT    Z9SYS1113CCATALOG.Z19.MASTER
SYSPARM   00
IEASYM    00
NUCLST    00
PARMLIB   USER.PARMLIB          Z9SYS1
PARMLIB   ADCD.Z19.PARMLIB      Z9RES1
PARMLIB   SYS1.PARMLIB          Z9RES1
NUCLEUS   1
SYSPLEX   ADCDPL
```

Change the 99 in the first line to 77 (or whatever number you used for your IODF). The format of this statement is odd, but it results in the name SYS1.IODF77. Do not change anything else in the LOADxx member unless you are certain about your actions.

The new IODF is now ready to use the next time you IPL z/OS with the parameter:

```
$ ipl 0a80 0a8200
```

Assuming you are satisfied with the results, you will probably want to change all the LOADxx members that you use. You must also change your devmap to use the new devices you added to your z/OS system.

(Note: At the time of writing, we have no information about the use of OSAD in the zPDT environment.)

## 3.9 Local printing

There is often less need for *hard copy* printed output in today's working environments, but it is sometimes needed. There are a variety of ways to approach this. The following material describes only one of these ways. This discussion assumes you are using a recent version of the AD-CD z/OS system.

### Background

Basic z/OS printing is closely related to the hardware available on the original S/360 machines. The most common printer at that time was the IBM 1403. It printed lines with 120 characters (or 132 characters, with an optional feature) and was normally set to print 6 lines per inch on fan-fold paper that was 11 inches long. This meant a full page held 66 lines. In practice, many programs counted output lines and skipped to a new page after 60 or 61 lines.

Much of the utility software with the system, such as JCL processors, assemblers, compilers, system report programs, and so forth were designed to fit these pages. That is, they printed lines of up to 120 characters (sometimes up to 132 characters) with about 60 lines per page. This default convention is still with us today.

Later hardware replaced the line printers (such as the 1403) with laser printers. These were devices such as the IBM 3800, 3820, 3825, 3900, and so forth. These could accept a variety of paper sizes, but were most commonly used with "letter size" paper.<sup>7</sup> With proper programming, these printers can produce sophisticated output using many fonts and graphics components. However, the system utilities (compilers, for example) continued to produce listings in "1403 format." Software for these laser printers can accept this 1403-format data and list it. These listings typically are two-sided, landscape mode, and contain up to 66 lines of 132 characters on each page.

Real 1403 printers are historical items, but there are many uses for pseudo-1403 devices. z/OS and JES2 still support 1403 printers and the 1090 can emulate 1403 printers. This is all that is needed for printed output from utilities, compilers, and many existing applications.

### Using a PC printer

Our goal was to use a common PC laser printer and have utility output produced in the format just described: landscape mode, 66 lines per page, 132 characters per line. If the PC printer provides duplex printing (printing on both sides of the paper), this would be used. The flow is illustrated in Example 3-2.

Our tests used a Lexmark OptraS1250 and Optra L printers (both with duplex printing features). We have not tried the techniques described here with other printers, but we expect the same or similar techniques could be used. However, remember that z/OS printed output

<sup>7</sup> The "letter size" (or A4 elsewhere) paper could be cut sheets or fanfold paper, depending on exactly which printer is being used.

typically contains separator pages, JCL listings and messages, and so forth; the smallest job usually has multiple pages of printed output. This may not be suitable for use with a small inkjet printer. We assume the use of a fairly heavy-duty laser printer for z/OS printing.

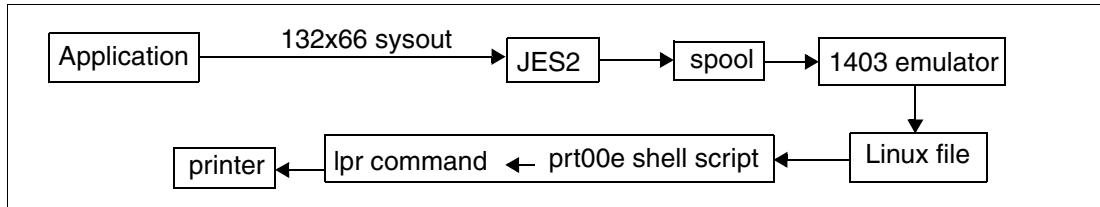


Figure 3-2 General flow for printing

### 3.9.1 Setup

We need to provide an appropriate setup for Linux, the 1090 devmap, a shell script, and JES2.

#### Linux setup

We first configured our (very old) Lexmark Optra S1250 for Linux. This printer has a parallel input; our computer had no parallel ports. We purchased a USB-to-parallel cable and this provided the needed connectivity. We used YAST (running under openSUSE 11.0) to configure the printer.<sup>8</sup> A print queue named `optras1250` was created automatically by YAST. We verified that the printer worked by using commands such as:

```
$ lpr /home/ibmsys1/prof10          (to print one of our devmaps)
```

#### Devmap setup

We added the appropriate 1403 definition to the 1090 devmap:

```
[manager]
name awsprt 4321 --windows
device 00E 1403 2821 /tmp/1403a
```

The `--windows` option is needed to place CR/LF characters in the output; without this option, NL characters are used and a PC printer may not be happy with NL characters. The output file name (`/tmp/1403a` in the example) is arbitrary. In our case, we did not expect much printed output and `/tmp` seemed a reasonable place to put it. The output file may also be assigned or changed with the `awsmount` command.

#### Shell script

We created a shell script named `prt00E` (the name is arbitrary) and placed it in our home directory (`/home/ibmsys1`, in our case). The shell script contained the following:

```
CTL="\033\105\033\050\163\060\160\061\066\056\066\067\150\070\056\166
\060\163\060\142\124\033\046\154\061\157\061\163\065\056\064\143
\055\061\060\060\060\132"
CTL2="\014\033\105"
(/bin/echo -ne $CTL; cat /tmp/1403a; /bin/echo -ne $CTL2) | lpr -P optras1250
```

The CTL and CTL2 constants are printer control characters, written in octal.<sup>9</sup> The octal format was the most convenient for use in a shell script. If you have good script writing skills you can do this several ways. The particular control characters shown here are for the Lexmark printers we mentioned. Your printer may require different controls. If you are printing to the

<sup>8</sup> Red Hat Linux has a different configuration process, but the end results are about the same.

<sup>9</sup> CTL is shown as three lines here, but it is actually created as one long line containing 38 octal constants.

default Linux printer you do not need the `-P` parameter (and queue name) of the `lpr` command.

The logic in the shell script is simple. It sends data (via a *pipe* and `lpr`) to the queue we defined earlier. It sends the CTL string (using `echo`), then sends the print data from the output file we named in the `devmap` or `awsmount` command (using `cat`), and then sends the CTL2 string (using `echo`) to flush the printer buffer and reset the printer.

The CTL control string (for our Lexmark printers) resets the printer, switches to a fixed font, sets a small type size pitch, changes to 8 lines/inch, uses a Courier font, uses landscape, duplex printing, uses 5.4/48 line height, and a small line offset to better center the data. The only unique requirement is that the format must use *exactly* 66 lines per page in order to synchronize with the pages produced by the 1403 emulator.

In the following strings `\033` is the ESC character that is used to begin printer command strings, and this is shown as a bold-face **E** in the character equivalents of the string. The octal constants are the equivalent of the characters shown and the CTL string could be written with characters (except for the ESC byte). The CTL commands are:

OCTAL constant.....	Characters	Comment
<code>\033\105</code>	<b>EE</b>	Reset printer
<code>\033\050\163\060\160</code>	<b>E(s0p</b>	Use a fixed font,
<code>\061\066\056\066\067\150</code>	<b>16.67h</b>	with 16.67 inch character pitch
<code>\070\056\166</code>	<b>8.v</b>	with 8 lines/inch characteristics
<code>\060\163\060\142\124</code>	<b>0s0bT</b>	using upright, medium Courier
<code>\033\046\154\061\157</code>	<b>E&amp;1lo</b>	Use landscape format
<code>\061\163</code>	<b>1s</b>	with duplex printing, long edge
<code>\065\056\064\143</code>	<b>5.4c</b>	5.4/48 inch line height
<code>\055\061\060\060\060\132</code>	<b>-1000Z</b>	line offset

The CTL2 commands are:

```
\014\033\105                                Force last page, reset printer
```

The CTL string was produced after some experimentation. It works for our printers, but it may contain unnecessary elements. Notice that we hard-coded the file name (`/tmp/1403a`) in the shell script; more skilled users may want to make this a command-line variable.

## JES2 setup

Normal z/OS printing flows through JES2 and printers must be known to JES2. Recent AD-CD systems do not have a printer defined for JES2; we need to define a 1403 at address 00E for JES2. (We use device number 00E because it is the traditional address for a 1403 printer and because it is already defined in the AD-CD IODF.) Edit AD-CD PARMLIB member JES2PARM to contain the following line:

```
PRT(1) WS=(W,R,Q,PMD,LIM/F,T,C,P),UNIT=00E,CLASS=C
```

If you scroll through the existing JES2PARM (in the AD-CD system) you will find commented lines similar to this. You can insert a new line, as shown, or convert the commented lines into active lines.<sup>10</sup> The print class (CLASS=C) is arbitrary; we selected class C because nothing defaults to this class.

We added the printer definition to JES2 and re-IPLed z/OS. (There are various ways to do the same thing without the re-IPL.) We verified that the printer was online (`d u,, ,00E,1`) and then issued a JES2 command to start it (`$SPRT1`). Use a `$SPRT1` command as the response to requests to mount forms and so forth.

<sup>10</sup> If you convert the commented lines into active lines, be especially careful with the `/*...*/` comment indicators; be certain you remove matching pairs.

## 3.9.2 Operational technique

We then ran jobs that sent output to SYSOUT=C. For example,

```
//OGDEN1 JOB 1,OGDEN,MSGCLASS=C
// EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//SYSUT1 DD DSN=ADCD.S110.PARMLIB(IEASYS00),DISP=SHR
//SYSUT2 DD SYSOUT=*
```

After submitting this job, you should notice z/OS console messages about jobs sent to PRT1. If JES2 requests a setup function for the printer, reply \$SPRT1. If the emulated printer is started (for JES2), the printed output is immediately sent to the “printer.” As described here, this is file /tmp/1403a in Linux. Additional output (from multiple jobs) is simply added to the file. The emulated printer cannot distinguish where one job ends and the next begins; the JES2 separator pages are needed for this.

At some point you can stop the JES2 printer (\$PPRT1) and print the accumulated output under Linux. (You do not need to stop the JES2 printer if you are certain no additional output will be sent to it.) You should disconnect the output file (/tmp/1403a) from the emulated printer:

```
$ awsmount 00E -u
```

You then run the shell script:

```
$ cd /home/ibmsys1           (Directory containing the shell script)
$ ./prt00E                   (Execute the shell script)
```

The printer should begin printing output. Notice that the output includes all the job separator pages produced by JES2. When it finishes, you can use **awsmount** to provide an empty output file for the emulated printer:

```
$ rm /tmp/1403a              (Delete the old output file)
$ touch /tmp/1403a           (Start a new output file; same name)
$ awsmount 00E -m /tmp/1403a (Connect new output file)
```

We deleted the output file (assuming we do not want to print the same jobs again). We then recreated the same file (because the name is hard-coded in the shell script) and “mounted” it on the emulated printer, ready for more output.

If you stopped the JES2 printer, you need to start it again (\$SPRT1).

## 3.10 Enabling TSO users for OMVS

Current releases of the z/OS AD system provide “built-in” userids such as ADCDA, ADCDB, ADCDC, and so forth. For some releases these userids are not enabled for Unix System Services (also known as OMVS). You can enable these userids with the steps described here.

Logon to TSO as *ibmuser* or *adcdmst*. These two userids have RACF® SPECIAL authority. Issue the following command from the TSO READY prompt or from ISPF option 6:

```
alg test omvs(gid(100))
```

For current AD systems, the ADCDx userids are in RACF group TEST. This command assigns Unix System Services group ID 100 to the RACF group named *test*. OMVS users

must be a member of a valid OMVS group and this command creates the needed OMVS group. Group ID 100 is arbitrary and could be any unassigned group number.

Next, issue a command such as the following:

```
alu adcde omvs(uid(105) program(/bin/sh) home(/u/adcde))
```

In this example, we are enabling userid *adcde* for OMVS usage; this could be any one of the existing userids in the AD system. We are assigning uid 105 for this user. This number is arbitrary, provided it does not duplicate an existing OMVS uid number in your system. For simplicity, we often assign OMVS uids using this pattern:

USER	UID
adcda	101
adcdb	102
adcdc	103
adcdd	104
etc	etc

There is no requirement to use this pattern, but it is a convenient way to avoid duplicating numeric uids for the built-in users in the AD system.

The last step is to start OMVS, using a userid that is already valid for OMVS. This step is easier if the userid is a superuser (with numeric uid 0). Users *ibmuser* and *adcdmst* have this characteristic in recent AD releases. From the OMVS prompt issue commands such as:

```
mkdir /u/adcde
```

This command creates a Unix System Services home directory for user *adcde*. A home directory is needed in order to log into Unix System Services with the OMVS command.

Once the user is enabled for Unix System Services functions, one of the following methods can be used to begin using Unix System Services:

1. Logon to TSO (using a 3270 emulator) and enter an **omvs** command from the READY prompt or from ISPF option 6. This produces a common UNIX® shell environment, adapted to 3270 terminals.
2. Logon to TSO (using a 3270 emulator) and enter an **ish** command from ISPF option 6. This produces a UNIX shell that is unique to z/OS and 3270 terminals.
3. Assuming you have connectivity to z/OS TCP/IP from your PC Linux system, you can use a command (from a Linux command window) such as **telnet 10.1.1.2 1023** to connect directly to Unix System Services. The IP address in the command (10.1.1.2 in this example) must be the IP address of your z/OS TCP/IP. (It is not the IP address of the base Linux in your zPDT system.) Port 1023 in the AD system is used for connection to Unix System Services. This method of accessing Unix System Services is *not* adapted to 3270 terminals. It uses a basic UNIX-style terminal and *vi*, for example, can be used in this mode.

## 3.11 SYS1.LOGREC full

Maintaining SYS1.LOGREC is a normal z/OS system programmer's task. There is nothing unique to the 1090 or the AD distribution. Production installations, using larger System z machines, often keep ordered histories of LOGREC data and study any new material in LOGREC. It includes data about hardware and software failures, IPL statistics, volume usage statistics, and so forth.

Most 1090 users simply ignore SYS1.LOGREC until they receive messages that it is full. These messages do no harm, but it is a good idea to clear LOGREC when such messages are received. There is at least one job in the AD libraries to do this, but the job name (and library name) may change from time to time. The following is a job to clear SYS1.LOGREC. The particular format shown here is quite old and should be used exactly as shown.

```
//BILLCL JOB 1,OGDEN,MSGCLASS=X
// EXEC PGM=IFCEREP1,PARM='CARD'
//SERLOG DD DISP=SHR,DSN=SYS1.LOGREC
//DIRECTWK DD UNIT=SYSDA,SPACE=(CYL,5,,CONTIG)
//EREPT DD SYSOUT=*,DCB=BLKSIZE=133
//ACCDEV DD DUMMY
//TOURIST DD SYSOUT=*,DCB=BLKSIZE=133
//SYSIN DD *
    SYSUM
    ACC=Y
    ZERO=Y
/*
```

You may find slightly different jobs that perform the same function and any of these should be acceptable. Never attempt to “clear” SYS1.LOGREC by simply deleting and reallocating it, or by removing records with a text editor.

Note that you can edit IEASYSxx members (in PARMLIB) to say LOGREC=IGNORE to avoid the LOGREC full problems.

## 3.12 Lost MVS console

MVS does not like to lose its operator console (this is not unique to zPDT operation!). If you accidentally close the TN3270e session that contains the MVS operator console, you might try the following recovery.

First, simply try to re-establish the session. This may be easier if the MVS console has an LU name in the devmap. For example,

```
$ x3270 -port 3270 mstcon@localhost &
```

Depending on exactly what was happening when the console was lost, the TN3270e connection to the aws3174 device manager may still be active and you cannot make a “new” connection to it. You can force this to completely disconnect by this command in a Linux window:

```
$ awsmount 700 -d (assuming your MVS console is address 700)
```

You might then be able to connect the TN3270e session to address 700. You need to have the TN3270e session connected before proceeding with additional recovery.

When MVS lost the console it may have started issuing messages in the Linux window used to start zPDT. These are “HMC hardware console” messages. You can attempt to reactivate the MVS console on 700 (assuming you have a TN3270e connection to 700) as follows:

```
$ oprmsg v cn(*),activate (activate the "hardware console" for commands)
$ oprmsg v 700,offline
$ oprmsg v 700,offline,force (if the simple vary offline fails)
$ oprmsg v 700,online
$ oprmsg v 700,console
$ oprmsg v cn(*),deactivate (optional)
```

This may not always work. Also, it may produce a console in 3270-2 (24 lines) mode, but this is better than no console.

### 3.13 Unable to start ISPF

When logging onto TSO, the following message is sometimes seen when attempting to start ISPF:

```
%% UNABLE TO ALLOCATE OR CREATE ISPF PROFILE DATASET
ISPF003 FOLLOWING FILE WAS NOT PREALLOCATED
ISPPROF
```

The most common reason for this problem is that the required ISPF profile data set was uncataloged for some reason. (Crashing z/OS at just the wrong moment can do this.)

You need to recatalog the profile data set. Assuming you are logging on as IBMUSER, the data set you want (in all recent AD-CD releases) is IBMUSER.ISPF.ISPPROF. You can logon with another userid (ADCDMST is convenient for this purpose) and recatalog the data set. The easiest way to recatalog the data set is to list the volume (SASYS1 or ZBSYS1, for example) using ISPF 3.4 and use a C line command to catalog the data set. (The “C” is entered at the beginning of the line for that data set in the ISPF 3.4 display.)

Another solution, after ISPF fails to start, is to preallocate the data set and then start ISPF again. The TSO command to do this is:

```
READY alloc da('ibmuser.ispf.ispprof') f(ispprof) shr vol(zbsys1) unit(3390)
READY ispf                               (start ISPF again)
```

You need to use the volser that matches your current system, of course.

### 3.14 Health Checker

The z/OS Health Checker has been included in recent AD-CD releases, but is not configured for operation. A quick configuration (z/OS 1.10 and 1.11) involves the following steps:

- ▶ Edit SYS1.SAMPLIB(HZSALLCP) and change the DSN operand (in the DD statement) to something like DSN=SYS1.ADCD.HZSPDATA. A VOL=SER= parameter may be added although the data set involved is not very large. Run this job.
- ▶ Edit SYS1.SAMPLIB(HZSPROC) and change the DSN operand to match that in the first step. Then copy this procedure member to the ADCD PROCLIB as member HZSPROC.
- ▶ Edit the ADCD PARMLIB member ISFPRM00 and add a CK parameter to the list of authorizations allowed for systems programmers. (Remember the comma!)
- ▶ Enter the following commands in the ISPF option 6 display:

```
RDEFINE XFACILIT HZS.* UACC(READ)
SETROPTS RACLIST(XFACILIT) REFRESH
RDEFINE STARTED HZSPROC STDATA(USER(START1) GROUP(SYS1) TRUSTED(YES))
SETROPTS RACLIST(STARTED) REFRESH
```
- ▶ From the MVS operator console, stop the SDSF server (**P SDSF**) and start it again (**S SDSF**).
- ▶ From the MVS operator console, start the Health Checker (**S HZSPROC**).
- ▶ Go to the SDSF primary panel and enter the **CK** command. This should display output from the Health Checker.

- ▶ Remember to stop it (P HZSPROC) when stopping z/OS. You might add the stop command to the SHUTDOWN scripts in the ADCD PARMLIB.

## 3.15 Reply needed to stop zFS

The AD-CD SHUTDOWN scripts contain the command:

```
f omvs,stoppfs=zfs
```

Changing this line to:

```
f omvs,shutdown          (through z/OS 1.10 only)
```

causes zFS to end without requiring an operator reply. (It also seems to take a little longer to end this way.)

## 3.16 RMF-III

Recent releases of the AD system may not start RMF-III correctly, with a failure finding a module. This can be corrected by adding GDDM.SADMMOD to the link list. (Consider adding it to all the PRODxx members in PARMLIB.)

## 3.17 Compressing PARMLIB

We typically make many PARMLIB changes (usually to the current ADCD PARMLIB) while adjusting the AD system to our individual needs. We then need to compress PARMLIB to recover space and prevent unexpected expansion into multiple extents. With recent AD-CD releases, a simple compress (using the Z option in the ISPF 3.4 panel) fails because the PARMLIB is being used by another job. Pressing PF1 twice (when this error message is received) displays the name of the conflicting job(s). The conflict may be from zFS. The following MVS console command:

```
f omvs,stoppfs=zfs
```

stops zFS and permits compression of the PARMLIB. This should be done when there are no Unix System Services users, of course.

## 3.18 SVC dumps for IRARMCPU

In rare cases, z/OS under zPDT may experience repeated 0C9 ABENDs and SVC dumps for module IRARMCPU. We have observed this only under prolonged very heavy CP loads (such as encountered installing WAS). z/OS recovers from the ABENDs, but the overhead of each SVC dump is substantial (and may consume considerable disk space).

Do not make the PARMLIB adjustments described here unless you experience this problem.

This is a z/OS problem that was observed in z/OS releases 1.9 and earlier distributions of 1.10 and is addressed by APAR OA27453. The exact conditions for the problem may occur only on zPDT. The error conditions may be avoided by setting a parameter in the IEAOPTxx member of PARMLIB. The IEAOPTxx member is referenced only if OPT=xx is included in the

IEASYSxx member of PARMLIB, and this is not present in the AD-CD IEASYSxx members. Assuming we are using IEASYS00, the bypass might be as follows:

```
PARMLIB member IEASYS00
...
OPT=00,                (Remember the comma !)
...

PARMLIB member IEAOPT00
...
BLWLTRPCT=0           (Comma needed if more parameters)
```

A typical AD system uses several IPL parameters, corresponding to several IEASYSxx members in PARMLIB. The OPT=xx parameter would be inserted into all the IEASYSxx members that are used. We stress that very few zPDT systems will need this bypass.

The default IEAOPT00 member in PARMLIB (which is not used unless OPT=00 is added to the IEASYSxx members) contains ERV=500 as the only parameter. We have retained this value during informal use of our system, but have seen suggestions that something like ERV=5000 would be a better choice.

## 3.19 Burning 3390 volumes on CD

If we wanted to preserve a 3390 volume on CD (or DVD), we could use the following command to make a compressed copy:

```
$ gzip -c /z/Z5RES1 > /z/Z5RES1.gz
```

We could then burn the compressed file on CD or DVD by using a normal Linux CD/DVD burning application.

## 3.20 Delete logstreams

The default location for log streams (for the AD-CD system) is the ZASYS1 volume (or the equivalent, for other releases). These streams can sometimes grow to fill the volume. A typical job for deleting a log stream is as follows:

```
//BILL1 JOB 1,OGDEN,MSGCLASS=X
// EXEC PGM=IXCMIAPU
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
  DATA TYPE(LOGR) REPORT(YES)
  DELETE LOGSTREAM NAME(WAS.ERROR.LOG)
/*
```

## 3.21 Installing a DEMOpkg

The DEMOpkg is a preinstalled z/OS system that was available for IBM internal use until recently. It was not otherwise available due to license issues. (New versions of the DEMOpkg are not available. The version based on z/OS 1.9 was the last one produced.)

The first DVD of the DEMOpkg contains several reference and usage guides. We strongly suggest reviewing these and perhaps printing them.

The DEMOpkg emulated 3390 volumes are in compressed tar format. When untarred they are placed in fixed subdirectories of the current Linux directory. Be certain you are in the desired directory before starting installation. The titles of the DVDs on the z/OS 1.9 DEMOpkg are zDEMOpkg1of8, and so forth. For our example, we installed the volumes on a USB disk drive.

```
(insert the first DVD)
$ cd /media/disk                               make USB drive the current directory
  (The USB drive is not necessary, of course. If you want to place the
  emulated volumes in /z, you would issue cd /z at this point.)
  (Verify your DVD title, shown here as zDVD1)
$ tar -xzvf /media/zDVD1/DMTRES.tgz           IPL volume
$ tar -xzvf /media/zDVD1/DMTCAT.tgz           IODF, catalog volume
$ tar -xzvf /media/zDVD1/DMTOS1.tgz
$ tar -xzvf /media/zDVD1/DMTOS2.tgz
$ tar -xzvf /media/zDVD1/DMTOS3.tgz
$ tar -xzvf /media/zDVD1/DMTOS4.tgz
$ tar -xzvf /media/zDVD1/DMTPG1.tgz           whole volume for paging
$ tar -xzvf /media/zDVD1/DMTSP1.tgz           whole volume for spool
```

This results in the emulated volumes being placed in /media/disk/zdemopkg/zOS/. The zdemopkg and zOS subdirectories are created automatically. We created the following devmap:

```
$ cd /home/ibmsys1
$ touch ademo1
$ gedit ademo1
  [system]
  memory 1600m
  processors 1
  3270port 3270

  [manager]
  name aws3274 0002
  device 463 3279 3274 mstcon
  device 470 3279 3274 tso

  [manager]
  name awsckd 0001
  device 1c0 3390 3990 /media/disk/zdemopkg/zOS/DMTRES
  device 1c1 3390 3990 /media/disk/zdemopkg/zOS/DMTCAT
  device 1c2 3390 3990 /media/disk/zdemopkg/zOS/DMTOS1
  device 1c3 3390 3990 /media/disk/zdemopkg/zOS/DMTOS2
  device 1c4 3390 3990 /media/disk/zdemopkg/zOS/DMTOS3
  device 1c5 3390 3990 /media/disk/zdemopkg/zOS/DMTOS4
  device 30c 3390 3990 /media/disk/zdemopkg/zOS/DMTPG1
  device 30d 3390 3990 /media/disk/zdemopkg/zOS/DMTSP1

  [manager]
  name awsosa 0009 --path=F0 --pathtype=OSD
  device 600 osa osa --unitadd=0
  device 601 osa osa --unitadd=1
  device 602 osa osa --unitadd=2
```

We used the following startup sequence:

```
$ awsstart ademol
$ x3270 -port 3270 localhost &          starts 463, the MVS console
$ x3270 -port 3270 localhost &          starts net 3270, device 470
$ ipl 1C0 parm 01C1DPM
```

We noted that the x3270 sessions operated only in 24-line mode, probably due to the VTAM® parameters in the z/OS TCPIP PROFILE. Use the DP parameter shown here if you IPL with the minimal system. The initial TSO userid is SYSPRG1 (and the password is SYSPRG1). A certain amount of customization should be done during the first IPL; this is explained in the DEMO documentation.

**Important:** Note that the addresses (device numbers) used by the AD system and the DEMOpkg are completely different. Your devmap must match the addresses of whichever system you are using. For example, the AD system has the MVS console at address 700, while the DEMOpkg has the MVS console at address 463.

## 3.22 Disabled waits

You may sometimes see a message in your zPDT Linux window such as:

```
Warning! Disabled Wait CPU 0
$ d psw                                <--you may issue this command
PSW for CPU 0 000A0000 00000xxx          (24/32 bit mode)
PSW FOR cpu 0 00020000 00000000 00000000 00000XXX (64-bit mode)
```

When you receive such a message, you may issue a command to display the PSW. The last three characters of the PSW should contain a wait state code; an extended code may be also present in other characters of the PSW. The following list provides abbreviated information about standard wait state codes. Please see the System Codes manual for more complete information about each code. Note that some wait states are restartable; again, see the System Codes manual for more information.

Partial list (see the notes at the end of the list)

- 002 - During IPL, an I/O operation was not initiated (1)
- 003 - IPL cannot continue; subchannel is not operational for IPL or IODF device (1)
- 004 - During initialization, I/O not initiated (1)
- 005 - I/O interruption during IPL and unit check (2)
- 006 - I/O error during IPL processing; SYSRES or IODF volume (2)
- 007 - During initialization console not available (3)
- 009 - System build error. (z/OS problem; should be rare)
- 00A - Cannot find SYS1.LINKLIB or SYS1.CSSLIB in catalog (4)
- 00B - Master scheduler abended (4)
- 00D - Master scheduler abended (4)
- 00E - Problem on SYSRES volume (SYS1.NUCLEUS) (2)
- 00F - IPL volume does not contain IPL text (5)
- 013 - Error during NIP (6)
- 014 - Recursive program checks (6)
- 017 - Unit check during IPL (2,5)
- 019 - IPL program in error (6)
- 01B - SLIP requests wait (7)
- 01C - Recursive abend in FRR (8)
- 020 - Reconfiguration initialization failed (9)
- 022 - Page fault - devices quiesced or not ready (10)
- 023 - Trace initialization failed (11)
- 025 - Duplicate entry point in nucleus (6)

02E - ASM detected too many I/O errors (10)  
 030 - ABEND during NIP (6)  
 031 - No UCB for SYSRES (1, 2)  
 032 - NIP module missing (6)  
 033 - I/O error in BLDL during NIP (6)  
 035 - Could not find entry point in nucleus (6)  
 037 - DSCB for SVCLIB, PARMLIB, LINKLIB could not be read (12)  
 038 - Not enough main storage (11)  
 039 - DASD mount conflict (1,13)  
 03A - Error building LPA (6)  
 03B - Required module is not in LPA (6, 14)  
 03C - ASM found not enough paging storage (15)  
 03D - Error building page tables (11)  
 03E - Not enough page slots to back master scheduler initialization (15)  
 03F - NIP function invoked incorrectly (6)  
 040 - ABEND during NIP (6)  
 044 - Machine check during NIP (6) (Try IPLing again, at least once!)  
 045 - NIP could not initialize RTM (6)  
 046 - Program check during NIP (6)  
 04A - TOD clock in error (16)  
 050 - Alternate CPU recovery (ACR) entered recursively (16, 6)  
 051 - ACR had error (software) (6)  
 052 - ACR error (hardware) (16)  
 053 - PC or PC/AUTH failed (17)  
 054 - Error in member loaded into nucleus (6,14)  
 055 - IPL cannot find necessary member in SYS1.NUCLEUS (14)  
 056 - NIP error (6)  
 059 - Unidentified return code for BLDL during NIP (6)  
 05A - ACR tried to remove last CP (16)  
 05C - NIP cannot find catalog pointer in nucleus (18,12)  
 05D - During NIP, could not find DSCB for catalog (12)  
 05E - Error reading master catalog (2)  
 060 - ASM detected errors in page tables (6)  
 061 - TOD clock errors during STCK instruction (16)  
 062 - Channel path error (16)  
 063 - NIP storage problem. SQA too small? (4)  
 064 - NIP error and RTM not initialized (6)  
 065 - NIP issued type 3 or 4 SVC before they were available (6)  
 06F - I/O problem - unusual (1,2,10,13)  
 070 - NIP: insufficient contiguous main storage (6,11)  
 071 - System or operator initialized a restart  
 072 - No more workspace for IPL (6,11)  
 073 - IPL program waiting for I/O or external interrupt (16)  
 074 - IPL program contains a logic error (6)  
 075 - IPL program could not load a module (4,14,6)  
 076 - IPL found non-fullword relocatable address constant (6)  
 077 - SVC entry point cannot be resolved (6)  
 078 - Master catalog could not be opened (2,4,6)  
 07B - Required processor facility not available (19)  
 07C - Initialization error, configuration problem. (18,1,see Codes manual)  
 07D - IEASYSxx PARMLIB member is bad; in error (4)  
 07E - Unable to obtain LSQA storage for SVC (11,6)  
 081 - SYS1.NUCLEUS occupies more than one extent  
 082 - System joining sysplex needs maintenance  
 083 - Incorrect address in PSA (16)  
 084 - RTM error (17,6)  
 085 - ASM warm start problem (1,10,20)  
 087 - System removed from sysplex (normal situation)  
 088 - IPL: error in LOADxx or NUCLSTxx (18,4,6)  
 089 - NIP found an error in a UCB (6)

08A - WTO error going to wait state  
 08C - WLM has recurring error (6)  
 08E - SYSEVENT error (6)  
 08F - Failure rebuilding work queues (6)  
 09x - SPINLOOP problem (See Codes manual for more information)  
 0A1 - Excessive SPINLOOP unresolved (21)  
 0A2 - XCF encountered cross system problem  
 0A3 - Unable to join global GRS  
 0A4 - ETR problem  
 0A5 - HCD problem (remember: zPDT does not support dynamic reconfiguration)  
 0A7 - Insufficient ESQA or ECSA storage (11)  
 0B0 - Could not recognize IODF specified in LOADxx (18,1)  
 0B1 - LOADxx member problem (18,1)  
 0B2 - No devices in IODF (18, or you created a bad IODF)  
 0B3 - Incorrect information in IPL parameter  
 0B4 - UIM specified unidentified device number (22)  
 0E0 - SIGNAL failed during NIP (16)  
 0E1 - SIGP STOP failed because processor was not operational (16)  
 0E3 - Insufficient virtual storage to initialize CSA (4)  
 0E8 - During NIP, the machine check handler failed (6)  
 101 - Program in supervisor state requested too much SQA (6)  
 102 - Program in supervisor state requested more pages of SQA than available (6)  
 104 - While processing ABEND SVC, program check occurred recursively (6)  
 110 - System detected hot I/O from device other than DASD (16; note MVS console messages)  
 111 - System detected hot I/O on DASD device (16, note MVS console messages)  
 112 - System detected hot I/O on reserved DASD (16, note MVS console messages)  
 113 - Failure during channel path recovery (16)  
 114 - Previous error affected SMP operation. See manual.  
 115 - DASD containing paging dataset is unavailable. (10)  
 116 - During restart, detected missing interrupt for paging device (16)  
 11A - Error during SVC 26 (6)  
 201 - Failure while creating COMMTASK (6)  
 202 - During system initialization, creation of console communications failed (6)  
 204 - Error during allocation (6)  
 205 - Attempted to load a module that was not in LINKST (6)  
 206 - Sysplex initialization operator message prompt  
 5C7 - Error during processor or system termination (6)  
 A00 - DAT error for system address space (16,6)  
 A01 - Error on only online processor (16)  
 A18 - Unsolicited Device end on paging volume; AVR failed. (complex)  
 A19 - Can no longer perform I/O (16)  
 A1E - Time-of-day clock failed (16)  
 A1F - Processor controller not available; TOD sync cannot occur (16)  
 A20 - System found page in FLPA that is not fixed (6)  
 A21 - Segment table entry for MLPA, PLPA, FLPA, or xFLPA is incorrect (6)  
 A22 - Error (probably hot I/O) invoked disabled console communication facility (see manual)  
 A23 - Program check during machine check handling on only online processor (16)  
 A24 - Loop while running machine check handler on only online processor (6,16)  
 A26 - Machine check on only online processor; interruption code incorrect (16)  
 A27 - Problems during machine check interruption handling  
 A28 - DAT-off machine check handler cannot start DAT-on machine check handler (6,16)  
 A29 - Problems stopping processor after program or machine check (6)  
 A2A - System detected LPA page that is not on paging data set (6)  
 A2B - Error in extended storage (16)  
 A70 - Console unavailable during NIP (3)  
 A71 - Reconfiguration problem (9; see codes manual)  
 A7A - Service processor interface failed (16,9)  
 B01-B1D Wait states used by the 3203/3211 utility

B20-B24 Wait states used by the stand-alone IOCP program  
 CCC - Wait state generated by QUISECE command  
 D0D - SMF had resource shortage (too much SMF output requested? Memory too small?)  
 E02 - Should never happen (16)  
 EC7 - Severe error in Unix System Services (6)  
 FFx - Non-IBM program created a wait state

The suggested actions provided by the following notes assume you are using z/OS (probably the AD-CD system) in a normal manner. That is, we assume that:

- ▶ You have not modified the system.
  - ▶ You are not working with authorized programs or code.
  - ▶ You have not installed middleware that operates as authorized code.
  - ▶ You are not actively disrupting the hosting Linux environment for zPDT.
1. Check that your devmap contains the necessary volumes and that they are at addresses supported by the IODF of your z/OS system. For example, with the AD system have your 3390 volumes addresses in the range AAO -AEF. Restart zPDT and try again. If necessary, verify your emulated 3390 volumes (using an `a1cckd xxxx -rs` command.)
  2. Verify that your IODF points to the correct volumes for SYSRES and the IODF volume. With the typical AD system these are addresses A80 and A82. Possibly these volumes are corrupted. Verify your emulated 3390 volumes (using an `a1cckd xxxx -rs` command), restart zPDT and try to IPL again.
  3. z/OS wants a NIP console. This is address 700 in existing AD systems. Be certain a 3270 session is connected to this address (or whatever NIP console address is specified for *your* z/OS system). Also be certain the Linux window you used to start zPDT is open, as this could provide an alternate, limited NIP console function in some cases. If a 3270 session at address 700 is available, try to IPL again.
  4. Have you altered a working system? Changed key PROCLIB members? Changed the catalog line in the LOADxx member in SYS1IPLPARM? This is not recoverable. You must IPL another z/OS system (or restore volumes for this z/OS.)
  5. Did you IPL the correct volume? Verify your devmap and try again.
  6. Internal z/OS problem. Probably not your fault unless the volume is corrupted. Try IPLing z/OS again. You may need to IPL another z/OS system or restore volumes for this z/OS system.
  7. Someone set a SLIP trap. This is probably a user-caused wait and whoever set the SLIP trap should know how to proceed. System can be restarted.
  8. Typically a system error, but could be an error in a software product. Try IPLing z/OS again, possibly without starting recently installed middleware.
  9. You attempted a reconfiguration option that is not available for zPDT.
  10. Possible devmap or PC disk error or emulated 3390 corruption error. Stop zPDT and verify 3390 emulated volume formats with `a1cckd xxxx -rs` commands.
  11. Have you defined enough memory for your System z? Try increasing the memory size in your devmap and restart zPDT. (Consult zPDT documentation to understand the maximum System z memory definition recommended for your configuration.) Most zPDT users run z/OS with *at least* 800 MB System z memory defined.
  12. Did you IPL the correct volume? Has someone deleted data sets on this volume? Has someone deleted system data sets?
  13. Check your devmap carefully. Are two 3390 definitions pointing to the same file? Verify that the 3390 devmap entries point to the correct Linux files. Fix your devmap and restart zPDT.

14. Has someone deleted members in any system libraries? This is probably not recoverable. You need to IPL another z/OS system and possibly repair this z/OS system.
15. You may have started a large program (such as WAS) and you do not have enough space in your paging data sets. You might add paging data sets.
16. Restart zPDT. If the error persists, contact the zPDT support team.
17. Probably due to bugs in a software product. Were you starting a new product when this happened? Not recoverable. ReIPL and try again.
18. Has anyone modified SYS1.IPLPARM? Are you using a new LOADxx member in this library? If so, the new member has an incorrect catalog line. IPL with a "standard" load parameter (to use a "standard" LOADxx member).
19. See the zPDT documentation for information about what facilities and functions are available through zPDT. If you are unable to resolve the problem, contact the zPDT support team.
20. Try a cold start (CLPA).
21. Try restarting zPDT with fewer processors defined in your devmap. Contact zPDT support if the problem persists with fewer devmap processors defined than you have real CPUs in your PC.
22. Check your devmap. Devmap and IODF must have compatible device addresses. Are you attempting to use an unsupported device?



## **z/VM notes**

This chapter contains notes about installing z/VM 6.1 from a standard z/VM DVD distribution set, a description of a prepackaged z/VM 5.4 system prepared for zPDT usage, and a few notes about general z/VM usage.

Most z/VM users on zPDT would find either z/VM release, 5.4 or 6.1, acceptable. The 5.4 release has the advantage that it is already in zPDT format and is ready to IPL. The 5.4 release was available to authorized zPDT users at the time of writing, but will eventually be removed in favor of later releases.

## 4.1 Installing z/VM 6.1

We obtained a standard z/VM 6.1 package in DVD format. The following briefly outlines the installation steps we used for it. (We installed it under SLES 11 as the base Linux system.)

We first used the `alckd` command to create five 3390-3 volumes. These were created in our Linux `/z` directory with the following names: 610RES, 610W01, 610W02, 610PAG, and 610SPL. These are the volume serial names of the z/VM volumes and we used the same names as Linux file names.

We created a devmap with two 3270 terminals (at addresses 700 and 701) and the five 3390-3 volumes:

```
[system]
memory 3500m
3270port 3270
processors 1

[manager]
name aws3274 0002
device 0700 3279 3274 mstcon
device 0701 3279 3274

[manager]
name awsckd 0101
device 0200 3390 3990 /z/610RES
device 0201 3390 3990 /z/610W01
device 0202 3390 3990 /z/610W02
device 0203 3390 3990 /z/610PAG
device 0204 3390 3990 /z/610SPL
```

We mounted the z/VM DVD in the DVD drive, issued an appropriate `awsstart` command, started an x3270 session, and issued the following zPDT command (in the Linux terminal window):

```
$ ip1_dvd /media/610\ ga\ 3390/cpdvd/610vm.ins -c 700
```

The back slashes in this command are due to the blanks in the DVD title. Using the Linux tab key for command completion eases the entry of this command. The `-c 700` parameter causes zPDT to emulate a System z HMC 3270 session with the terminal at the indicated address.

Reply Y to the next message; this is a single character reply, not using the `oprmsg` function.

A memory-resident z/VM should then appear on the x3270 session. The remaining installation follows the *Guide for Automated Installation and Service*, GC24-6197, almost exactly. A user new to z/VM should obtain and follow the installation manual. The following brief outline is intended for experienced z/VM users who may need terse reminders about the installation process. The only unique zPDT element is the `mount_dvd` command used to mount the z/VM service (RSU) DVD.

Enter an `instplan` command (in the x3270 session). We took most of the default options, selecting 3390-3 devices for installation. The addresses entered must match the devmap; they are quite arbitrary, but we use addresses in the 200 range for our examples. Once the addresses were entered in the z/VM panel, followed by PF5, we attached these drives to our z/VM session:

```
attach 200 *
```

```
attach 201 *
attach 202 *
attach 203 *
attach 204 *
```

We then replied Y to the prompt for formatting them. This was followed by many messages generated during the installation process. Eventually we were prompted to mount the RSU DVD. We removed the z/VM installation DVD, mounted the RSU DVD, and entered the following zPDT command<sup>1</sup> (in the Linux terminal window):

```
$ mount_dvd /media/My\ CD/cpdvd/
```

and replied go. When prompted, we IPLed the new z/VM with:

```
$ ipl 200 parm 0700
```

which produced the stand-alone loader panel, and here we entered **cons=0700** followed by PF10. The initial z/VM commands are COLD DRAIN NOAUTOLOG. After the IPL completed, we disconnected the OPERATOR session and logged in as MAINT. We issued **i cms** followed by the command **instvm dvd**. This produced a long series of installation messages and ended with INSTVM EXEC ENDED SUCCESSFULLY.

We SHUTDOWN and IPLed again, and again switched to userid MAINT. We issued **i cms** and then **access 500 c**. On this volume we found file 6101RSU1 SERVLINK C1. We issued the command **SERVICE ALL 6101RSU1** and waited through many messages for **SERVICE PROCESSING COMPLETED SUCCESSFULLY**.

We then issued **access 5E6 B** followed by the command **PUT2PROD**. This produced many messages and ended with **PUT2PROD PROCESSING COMPLETED SUCCESSFULLY**.

We SHUTDOWN and IPLed again. At this point, we logged into AUTOLOG1 and added the command **CP ENABLE ALL**.

This completed our basic installation.

## 4.2 A prepackaged z/VM 5.4 system

A DVD containing a prebuilt z/VM 5.4 system<sup>2</sup> is available to 1090 users who are eligible for general AD-CD system distributions. The discussion and examples in this section are based on this particular z/VM system running on a 1090 system. The general z/VM concepts discussed should apply to any z/VM implementation, but the specific implementation and examples in this document are based on the z/VM 5.4 system on this DVD.vv.<< is tthis ok? vv?>>

Five 3390-3 volumes are used for our z/VM system:

- ▶ 540RES - The z/VM IPL volume (mounted at address 200 in our examples)
- ▶ 540W01 - Additional z/VM system minidisks (address 201)
- ▶ 540W02 - Additional z/VM system minidisks (address 202)
- ▶ 540PAG - z/VM paging space (address 203)
- ▶ 540SPL - z/VM spool space (address 204)
- ▶ PRODPK - Additional program products (address 205)

<sup>1</sup> This mounting of an RSU volume is the only current usage for the **mount\_dvd** command.

<sup>2</sup> This was true at the time of writing. A later z/VM version may be available when you read this.

z/VM usage is not bound to these addresses. However, we use these addresses in our discussions and 1090 devmap examples.

## 4.3 Installing the AD-CD z/VM 5.4 DVD

We assume the z/VM volumes are to be installed in a /z directory in Linux. (There is no requirement to do this, but it is compatible with setup recommendations in the other volumes in this series.) The DVD volume name is zVM54. The DVD contains six gzipped files; installation consists of unzipping these files into the /z directory. This may be done with the following Linux commands:

```
$ cd /media/zVM54
$ gunzip -c 540RES.gz > /z/540RES
$ gunzip -c 540W01.gz > /z/540W01
$ gunzip -c 540W02.gz > /z/540W02
$ gunzip -c 540PAG.gz > /z/540PAG
$ gunzip -c 540SPL.gz > /z/540SPL
$ gunzip -c PRODPK.gz > /z/PRODPK
```

### 4.3.1 1090 devmap

We created a devmap named devmapvm that defines a minimal z/VM system. We normally used **gedit** to edit devmaps but you may use any convenient editor. The devmapvm file is as follows:

```
[system]
memory 1600m
3270port 3270
processors 1

[manager]
name aws3274 0002
device 0700 3279 3274 mstcon
device 0701 3279 3274
device 0702 3279 3274

[manager]
name awsckd 0101
device 0200 3390 3990 /z/540RES
device 0201 3390 3990 /z/540W01
device 0202 3390 3990 /z/540W02
device 0203 3390 3990 /z/540PAG
device 0204 3390 3990 /z/540SPL
device 0205 3390 3990 /z/PRODPK
```

The addresses (disks at 0200, for example) are quite arbitrary. We placed the local 3270 terminals at 700 because we want to use an expanded version of this devmap with both z/OS, z/VM, and z/OS under z/VM.

### 4.3.2 Initial IPL

We used the following sequence for our initial IPL:

```
$ awsstart devmapvm
```

```
(wait until finished)
$ x3270 localhost:3270 &
$ x3270 localhost:3270 &
$ ipl 200 parm 0700
```

This should produce the Stand Alone Loader panel shown in Figure 4-1.

```
STAND ALONE PROGRAM LOADER: z/VM VERSION 4 RELEASE 4.0
DEVICE NUMBER: 0400      MINIDISK OFFSET: 00000000  EXTENT: 1

MODULE NAME:  CLOAD      LOAD ORIGIN: 1000

-----IPL PARAMETERS-----
cons=700

-----COMMENTS-----

9= FILELIST  10= LOAD  11= TOGGLE EXTENT/OFFSET
```

Figure 4-1 z/VM stand-alone loader

To complete the z/VM IPL, enter **cons=700** (as shown in Figure 4-1) and press F10. This should produce the display shown in Figure 4-2.

```
14:12:37 z/VM V5 R4.0 SERVICE LEVEL 0801 (64-BIT)
14:12:40 SYSTEM NUCLEUS CREATED ON 2008-07-29 AT 14:19:31, LOADED FROM 540RES
14:12:40
14:12:41 *****
14:12:41 * LICENSED MATERIALS - PROPERTY OF IBM* *
14:12:41 * * *
14:12:41 * 5739-A03 (C) COPYRIGHT IBM CORP. 1983, 2003. ALL RIGHTS *
14:12:41 * RESERVED. US GOVERNMENT USERS RESTRICTED RIGHTS - USE, *
14:12:42 * DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP SCHEDULE *
14:12:42 * CONTRACT WITH IBM CORP. *
14:12:42 * * *
14:12:42 * * TRADEMARK OF INTERNATIONAL BUSINESS MACHINES. *
14:12:42 *****
14:12:42
14:12:42 HCPZC06718I Using parm disk 1 on volume 540RES (device 0200).
14:12:42 HCPZC06718I Parm disk resides on cylinders 39 through 158.
14:12:43
14:12:43 HCPISU951I CP VOLID 440W03 NOT MOUNTED
14:12:43 HCPISU951I CP VOLID 440W08 NOT MOUNTED
14:12:43 Start ((Warm|Force|COLD|CLEAN) (DRain) (DIsable) (NODIRect)
14:12:43 (NOAUTOlog)) or (SHUTDOWN)
cold drain noautolog

CP READ ZVMV5R40
```

Figure 4-2 z/VM IPL panel

For our first IPL we entered **cold drain noautolog**, as shown in the figure. For subsequent IPLs, we just pressed Enter at this point.

After clearing the panel at least once, lines similar to those shown in Figure 4-3 should appear. At this point the OPERATOR userid is active on this panel. For the small, single-user z/VM we describe here, the OPERATOR userid is seldom needed.

```

18:00:28 Command accepted
18:00:28 XAUTOLOG DISKACNT
18:00:28 Command accepted
18:00:28 XAUTOLOG AUTOLOG1
18:00:28 Command accepted
18:00:28 XAUTOLOG OPERSYMP
18:00:28 Command accepted
18:00:28 AUTO LOGON ***      EREP      USERS = 2      BY OPERATOR
18:00:28 AUTO LOGON ***      DISKACNT USERS = 3      BY OPERATOR
18:00:28 AUTO LOGON ***      AUTOLOG1 USERS = 4      BY OPERATOR
18:00:28 AUTO LOGON ***      OPERSYMP  USERS = 5      BY OPERATOR
18:00:28 HCPCLS6056I XAUTOLOG information for AUTOLOG1: The IPL command is verified by the IPL command processor.
18:00:28 HCPCLS6056I XAUTOLOG information for EREP: The IPL command is verified by the IPL command processor.
18:00:28 HCPCLS6056I XAUTOLOG information for DISKACNT: The IPL command is verified by the IPL command processor.
18:00:28 HCPCLS6056I XAUTOLOG information for OPERSYMP: The IPL command is verified by the IPL command processor.
18:00:30 AUTO LOGON ***      VMSESVS  USERS = 6      BY AUTOLOG1
18:00:30 AUTO LOGON ***      VMSESVU  USERS = 7      BY AUTOLOG1
18:00:30 AUTO LOGON ***      VMSESVR  USERS = 8      BY AUTOLOG1
18:00:30 AUTO LOGON ***      DTCVSW1  USERS = 9      BY AUTOLOG1
18:00:30 AUTO LOGON ***      DTCVSW2  USERS = 10     BY AUTOLOG1
18:00:30 USER DSC LOGOFF AS AUTOLOG1 USERS = 9
18:00:30 AUTO LOGON ***      TCPIP    USERS = 10     BY AUTOLOG1
18:00:30 * MSG FROM OPERSYMP: 3 RECORDING FILE(S), 3 RECORDS, A DISK 01 % FULL
18:00:30 * MSG FROM EREP : 1 RECORDING FILE(S), 81 RECORDS, A DISK 04 % FULL
18:00:30 * MSG FROM DISKACNT: 4 RECORDING FILE(S), 246 RECORDS, A DISK 10 % FULL
18:00:30 HCPCRC8064I Recording data retrieval has been started; recording *ACCOUNT for userid DISKACNT.
18:00:31 HCPCRC8064I Recording data retrieval has been started; recording *SYMPTOM for userid OPERSYMP.
18:00:31 HCPCRC8064I Recording data retrieval has been started; recording *LOGREC for userid EREP.
18:00:31 OSA 0400 ATTACHED TO TCPIP 0001 BY TCPIP
18:00:31 OSA 0401 ATTACHED TO TCPIP 0002 BY TCPIP
18:00:31 OSA 0402 ATTACHED TO TCPIP 0003 BY TCPIP

Running      ITS01090
=====

```

Figure 4-3 Last OPERATOR panel after IPL

If a z/VM logo is not displayed on any other connected 3270 sessions, enter the command **enable all** on the OPERATOR panel.

Typical z/VM operation is to use the **disc** (disconnect) function for this session. This leaves OPERATOR running as a z/VM virtual machine, but with no console connected to it. We can log on to the OPERATOR user ID again later and resume its activities. (The password for userid OPERATOR is OPERATOR.)

The **disc** command should produce a logon panel similar to that shown in Figure 4-4.

```

z/VM ONLINE

                                     / VV          VVV MM          MM   5.4
                                     VV          VVV  MMM          MMM
      ZZZZZZ                          VV          VVV  MMMM        MMMM
***      ZZ                          VV          VVV  MM MM MM MM   ***
***      ZZ                          VV  VVV          MM  MMM  MM   ***
***      ZZ                          VVVVV          MM  M  MM     ***
      ZZ                              VVV          MM          MM|
ZZZZZZ /                             V           MM          MM|

      built on IBM Virtualization Technology

Fill in your USERID and PASSWORD and press Enter
(Your password will not appear when you type it)
USERID  ==>>
PASSWORD ==>>
COMMAND ==>>

Running ZVMV5R4

```

Figure 4-4 Logon panel

### 4.3.3 Initial usage

The VM user MAINT is the primary administration userid and has all initial authority in the system. In a large production operation, MAINT would be used sparingly.<sup>3</sup> In our small sandbox VM system, we use MAINT frequently. Enter MAINT as the userid and MAINT as the password in the logon panel. (All the VM user IDs<sup>4</sup> provided with a VM system have the passwords the same as the user names. There are a few exceptions where it is not possible to log on to certain user names.)

This should produce the results shown in Figure 4-5. The system will stop and display VM READ on the bottom right of the panel. Press Enter and the last line shown in the figure will be displayed. At this point, MAINT is logged on to the system and is running CMS.

```

LOGON MAINT
z/VM Version 5 Release 2.0, Service Level 0602 (64-bit),
built on IBM Virtualization Technology
There is no logmsg data
FILES: 0004 RDR, NO PRT, NO PUN
LOGON AT 17:35:51 EST THURSDAY 02/15/07
z/VM V5.2.0 2006-10-25 23:13

VM READ ITS01090
=====

```

Figure 4-5 After MAINT's logon

<sup>3</sup> MAINT is quite similar to IBMUSER in an initial z/OS installation and is somewhat like root in a Linux system. These user IDs have all the necessary authority to further customize and manage the systems. In production operations, various authorities are delegated to other user IDs, and the MAINT/IBMUSER IDs are seldom used. In small sandbox systems, MAINT/IBMUSER/root are often directly used, even when not strictly necessary.

<sup>4</sup> The terms “user” and “user ID” and “virtual machine name” are used interchangeably, although sometimes the use of “virtual machine name” implies that the corresponding user/user ID has logged on to the system to create the virtual machine.

This document is not intended as an introduction to z/VM usage. Considerable introductory material may be found in *Introduction to the New Mainframe: z/VM Basics*, IBM order number SG24-7316. As a quick refresher, we list here some of the CP and CMS commands that are immediately useful:<sup>5</sup>

- ▶ **i cms** - IPL CMS in your virtual machine, if it is not automatically IPLed. A **RUNNING** indicator at the bottom right side of the panel indicates that CMS is running.
- ▶ **q disk** - list your minidisks.
- ▶ **q da all** - list the “real” online disks.
- ▶ **q alloc all** - list page, spool, temporary disks, and directory usage.
- ▶ **q alloc map** - lists percentage use for page and spool disks.
- ▶ **q system** - another way to list system disks.
- ▶ **q accessed** - list minidisks I have accessed.
- ▶ **q links 120** - list userids who have links to my 120 disk.
- ▶ **q pf** - list Program Function Key assignments.
- ▶ **set pf12 retrieve** - make PF12 function as a retrieve key.
- ▶ **force userid** - terminate a user immediately.
- ▶ **q rdr** - list the files in your virtual reader.
- ▶ **rdrlist** - list files in your virtual reader.
  - Use PF11 to **peek** at any of these files.
  - Use **discard** to delete a particular file.
- ▶ **q prt** - list files in your virtual print queue.
- ▶ **purge system rdr all** - purge all reader files in the z/VM system.
  - **purge joe rdr 1234** - purge particular reader file belonging to userid joe.
  - **purge joe rdr all** - purge all joe’s reader files.
- ▶ **purge system prt all** - purge all printer files in the z/VM system.
- ▶ **link joe 456 456** - link to joe’s 456 disk as my 456 disk.
- ▶ **acc 456 j** - access my 456 disk as CMS drive j.
- ▶ **filelist \* \* a** - list the files on your a disk.
- ▶ **rel j** - release a CMS drive assignment
- ▶ **det 456** - detach disk 456 from my userid
- ▶ **vmlink joe 345** - a combined **link** and **acc** function.
- ▶ **format 191 a** - format a new minidisk.
- ▶ **x myfile text a** - (or enter **x** at the beginning of a filelist line) Start xedit.
  - The command line is at the bottom of the panel in our examples. (It can be positioned to the top of the panel, if desired.)
  - Important commands (on the command line) include:
    - **file** - Save the current file and exit.
    - **qquit** - Exit without saving the current file.
    - **/xxx/** - Locate the next line containing characters xxx.
    - **+nn** - Scroll forward nn lines.
    - **-nn** - Scroll backward nn lines.
    - **top** - Scroll to the beginning of the file.
    - **bottom** - Scroll to the bottom of the file.
    - **nulls on** or **nulls off** - Use the 3270 null character function.
  - Important default PF key definitions include:
    - PF2 - Insert a blank line after the line with the cursor.
    - PF3 - Quit (if no changes were made to the file); exit from the Help function.
    - PF4 - Tab to columns 5, 10, 15, and so forth.
    - PF7 - Scroll backward.
    - PF8 - Scroll forward.
    - PF9 - Split or join, depending on the cursor location.
    - PF10 - Scroll right 10 columns.
    - PF11 - Scroll left 10 columns.

---

<sup>5</sup> Some of the commands shown can be used only by privileged users, such as MAINT.

- PF12 - File.
- *Line commands* (as opposed to commands in the command line) are entered by typing over the numeric line number field at the left of each text line. (These line numbers are not part of the data record). Common line commands include:
  - **d** - Delete this line.
  - **i** - Insert a blank line after this line.
  - **"** - Duplicate this line (using the “double quotation marks” character).
  - **dnn** - Delete the indicated number of lines. This form can also be used with the **i** and **"** commands.
  - **dd** followed by **dd** in a later line - Delete a block of lines. This form can also be used with **"** **"** block commands to duplicate a block.
  - **cc** followed by **cc** in a later line - Copy a block of lines. The target line is noted by **p** (copy prior to this line) or **f** (copy following this line). An **mm** block can be used to move lines in the same way.
- ▶ **x user direct c** - edit the z/VM directory. Do this carefully!
- ▶ **directxa user direct c** - activate an updated z/VM directory.
- ▶ **ind** - how busy is the system?
- ▶ **q stor** - how much System z storage?
- ▶ **q n** - which userids are logged on?
- ▶ **q all** - what disks and terminals are online?
- ▶ **diskmap user direct c** - create a minidisk map based in directory *user direct c*.
- ▶ **browse user diskmap a** - inspect a file
- ▶ Update system configuration file - do this carefully:
  - **cprel a**
  - **link \* cf1 cf1 mr**
  - **acc cf1 e**
  - **x system config e**
    - edit as required
  - **acc 193 g**
  - **cpsyntax system config e**
  - **x local logo** - change the logo, if desired
  - **rel e**
  - **det cf1**
  - **cpaccess maint cf 1**

#### 4.3.4 AUTOLOG1 functions

We suggest logging onto userid AUTOLOG1 (password AUTOLOG1) and editing a file as follows:

```

x profile exec a                edit file profile exec a
add the following three lines at the end of the file
'CP ENABLE ALL'
'CP RECORDING ALL OFF'
'CP EXEC TELL OP RECORDING ALL OFF WAS EXECUTED'

```

This causes any connected 3270 sessions to be made available. These commands (in the profile for AUTOLOG1) are executed automatically when z/VM is started.

## 4.4 Multiple guest z/OS users

This description is based on the z/VM 5.4 system, but should be generally usable on any other z/VM system. In order to run two z/OS systems under z/VM, we added three userids to the z/VM system:

- ▶ MVSDUMMY - This userid owns the z/OS volumes. The use of a dummy owner is needed to allow the volumes to be shared by multiple z/OS guests
- ▶ ZOS1 - A userid for running z/OS. We typically used this to run the full AD system.
- ▶ ZOS2 - Another userid for running z/OS. We used this to run the stand-alone AD volume.

Remember, of course, that you cannot IPL the *same* z/OS system by multiple guests at the same time. This would cause z/OS to overwrite its paging data sets, among many other problems. (This statement ignores the case where different IPL parameters are specified to cause z/OS to initialize itself in different ways.)

We used the following devmap:

```
[system]
memory 3000m
3270port 3270
processors 2
cpuopt vmwait

[manager]
name aws3274 0002
device 0700 3279 3274 mstcon
device 0701 3279 3274
device 0702 3279 3274
device 0703 3279 3274
device 0704 3270 3274

[manager]
name awsckd 0101
device 0200 3390 3990 /z/540RES
device 0201 3390 3990 /z/540W01
device 0202 3390 3990 /z/540W02
device 0203 3390 3990 /z/540SPL
device 0204 3390 3990 /z/540PAG
device 0205 3390 3990 /z/PRODPK

[manager]
name awsckd 0001
device 0A80 3390 3990 /z/ZARES1
device 0A81 3390 3990 /z/ZARES2
device 0A82 3390 3990 /z/ZASYS1
device 0A83 3390 3990 /z/ZAUSS1
device 0A84 3390 3990 /z/ZAPRD1
device 0A85 3390 3990 /z/ZAPRD2
device 0A93 3390 3990 /z/SARES1
# You could include the rest of the AD volumes, if wanted

[manager]
name awsosa 0009 --path=F0 --pathtype=OSD
device 400 osa osa --unitadd=0
device 401 osa osa --unitadd=1
```

```

device 402 osa osa --unitadd=2

[manager]
name awsosa 0019 --path=A0 --pathtype=OSD --tunnel_intf=y
device 404 osa osa --unitadd=0
device 405 osa osa --unitadd=1
device 406 osa osa --unitadd=2

[manager]
name awstape 004
device 581 3490 3490
#device FFF 3490 3490

#[manager]
#name awscmd 3004
#device 580 3490 3490

```

This devmap can be used to run z/VM, or run z/OS, or run z/OS under z/VM. In our simple setup, under z/VM the two OSA channels are dedicated to userid ZOS1. This is intended as a *getting started* example. More extensive usage might use a VSWITCH link to OSA, or perhaps define more OSA devices in the devmap and define these for z/VM TCP/IP use.

We added the following entries in the z/VM directory:

```

*****
* BILL's ADDITIONS
*****
* YOU MAY DEFINE MORE MDISKS THAN IN YOUR DEVMAP
USER MVSDUMMY NOLOG 256M 256M G
MDISK A80 3390 DEVNO A80 MWV
MDISK A81 3390 DEVNO A81 MWV
MDISK A82 3390 DEVNO A82 MWV
MDISK A83 3390 DEVNO A83 MWV
MDISK A84 3390 DEVNO A84 MWV
MDISK A85 3390 DEVNO A85 MWV
MDISK A86 3390 DEVNO A86 MWV
MDISK A87 3390 DEVNO A87 MWV
MDISK A88 3390 DEVNO A88 MWV
MDISK A89 3390 DEVNO A89 MWV
MDISK A8A 3390 DEVNO A8A MWV
MDISK A8B 3390 DEVNO A8B MWV
MDISK A8C 3390 DEVNO A8C MWV
MDISK A8D 3390 DEVNO A8D MWV
MDISK A8E 3390 DEVNO A8E MWV
MDISK A8F 3390 DEVNO A8F MWV
MDISK A90 3390 DEVNO A90 MWV
MDISK A91 3390 DEVNO A91 MWV
MDISK A92 3390 DEVNO A92 MWV
MDISK A93 3390 DEVNO A93 MWV
MDISK A94 3390 DEVNO A94 MWV
MDISK A95 3390 DEVNO A95 MWV
MDISK A96 3390 DEVNO A96 MWV
MDISK A97 3390 DEVNO A97 MWV
*
USER ZOS1 ZOS1 1000M 1200M G
MACH ESA

```

```

CPU 00
SPOOL 00C 2540 READER *
SPOOL 00D 2540 PUNCH
SPOOL 00E 1403 A
CONSOLE 0700 3215 T
LINK MAINT 0190 0190 RR
LINK MAINT 019D 019D RR
LINK MAINT 019E 019E RR
SPECIAL 0701 3270
SPECIAL 0702 3270
SPECIAL 0703 3270
SPECIAL 0704 3270
DEDICATE 400 400
DEDICATE 401 401
DEDICATE 402 402
DEDICATE 404 404
DEDICATE 405 405
DEDICATE 406 406
MDISK 191 3390 2371 05 540W02 MR <==read text before using this statement
LINK MVSDUMMY A80 A80 MW
LINK MVSDUMMY A81 A81 MW
LINK MVSDUMMY A82 A82 MW
LINK MVSDUMMY A83 A83 MW
LINK MVSDUMMY A84 A84 MW
LINK MVSDUMMY A85 A85 MW
LINK MVSDUMMY A86 A86 MW
LINK MVSDUMMY A87 A87 MW
LINK MVSDUMMY A88 A88 MW
LINK MVSDUMMY A89 A89 MW
LINK MVSDUMMY A8A A8A MW
LINK MVSDUMMY A8B A8B MW
LINK MVSDUMMY A8C A8C MW
LINK MVSDUMMY A8D A8D MW
LINK MVSDUMMY A8E A8E MW
LINK MVSDUMMY A8F A8F MW
LINK MVSDUMMY A90 A90 MW
LINK MVSDUMMY A91 A91 MW
LINK MVSDUMMY A92 A92 MW
LINK MVSDUMMY A93 A93 MW
LINK MVSDUMMY A94 A94 MW
LINK MVSDUMMY A95 A95 MW
LINK MVSDUMMY A96 A96 MW
LINK MVSDUMMY A97 A97 MW
*
USER ZOS2 ZOS2 800M 1200M G
MACH ESA
CPU 00
SPOOL 00C 2540 READER *
SPOOL 00D 2540 PUNCH
SPOOL 00E 1403 A
CONSOLE 0700 3215 T
LINK MAINT 0190 0190 RR
LINK MAINT 019D 019D RR
LINK MAINT 019E 019E RR
SPECIAL 0701 3270

```

```

SPECIAL 0702 3270
SPECIAL 0703 3270
SPECIAL 0704 3270
MDISK 191 3390 2376 005 540W02 MR  <==read text before using this statement
LINK MVSDUMMY A80 A80 MW
LINK MVSDUMMY A81 A81 MW
LINK MVSDUMMY A82 A82 MW
LINK MVSDUMMY A83 A83 MW
LINK MVSDUMMY A84 A84 MW
LINK MVSDUMMY A85 A85 MW
LINK MVSDUMMY A86 A86 MW
LINK MVSDUMMY A87 A87 MW
LINK MVSDUMMY A88 A88 MW
LINK MVSDUMMY A89 A89 MW
LINK MVSDUMMY A8A A8A MW
LINK MVSDUMMY A8B A8B MW
LINK MVSDUMMY A8C A8C MW
LINK MVSDUMMY A8D A8D MW
LINK MVSDUMMY A8E A8E MW
LINK MVSDUMMY A8F A8F MW
LINK MVSDUMMY A90 A90 MW
LINK MVSDUMMY A91 A91 MW
LINK MVSDUMMY A92 A92 MW
LINK MVSDUMMY A93 A93 MW
LINK MVSDUMMY A94 A94 MW
LINK MVSDUMMY A95 A95 MW
LINK MVSDUMMY A96 A96 MW
LINK MVSDUMMY A97 A97 MW
*
* END OF BILL's ADDITIONS
*****

```

It is not possible to log on to MVSDUMMY. (Among other factors, the NOLOG in the user definition prevents this.) The format of the MDISK definitions used here is for full-pack minidisks. These point to volumes that are exactly the same as those used for z/OS without z/VM. We can IPL z/OS directly using these same volumes, without involving z/VM. The following directory statement says that this virtual machine's A80 device is a real 3390 at real address (as defined by the 1090 devmap) A80:

```
MDISK A80 3390 DEVNO A80 MWV
```

We could have different virtual and real addresses, but there is no reason to do this here, and it would only confuse the situation. The MWV operand indicates this is a multi-user read/write volume and that channel *reserve* and *re/lease* commands are to be emulated.

The purpose of some of the directory definition statements is as follows:

- ▶ The USER statement provides the virtual machine name (ZOS1) and the password (also ZOS1), indicates it is to have 1000 MB main storage, and is to have only G class z/VM command authority. (This is the authority of a routine z/VM user and is all that is needed.)
- ▶ The MACH ESA statement indicates the mode of the emulated machine. The CPU statement is optional when only a single processor is involved.
- ▶ The SPOOL, CONSOLE, and first set of LINK statements are normal for a CMS user. (The LINK TCPMAINT 592 statement allows the use of TCP/IP functions while in CMS.) We do not require this user to have CMS capabilities, but it does no harm and might be useful in some circumstances. z/OS running in this virtual machine does not normally use any of these definitions. The CONSOLE address is set to 700, because we will use the same

terminal address as the MVS console, and this needs to be address 700 for the z/OS system we used.

- ▶ The DEDICATE statements assign real devices to this guest machine.
- ▶ The SPECIAL statements define virtual local 3270 terminals at the indicated virtual addresses. These appear to z/OS as local, channel-attached 3270s. A user with access to a z/VM 3270 can use a **dial zos1** command to connect to one of these virtual 3270s. Our z/OS has 3270s at these addresses connected to VTAM, and a user will receive the VTAM logo panel when the user dials one of these terminals.
- ▶ The MDISK statement provides the A minidisk for CMS. Again, we do not require CMS for this user ID, but it might be useful in some circumstances. Be certain not to use conflicting cylinder addresses if you later add more minidisks.
- ▶ The remaining LINK statements provide access to the disk drives defined by the MVSDUMMY user ID. The drive defined as address A80 in MVSDUMMY is also addressed as A80 by this user, and so forth. The MW operand indicates the drive is used by multiple users in read/write mode.

The following MDISK statement creates a CMS “A disk” for the user ZOS1:

```
MDISK 191 3390 2371 10 540W02 MR
```

This minidisk starts at cylinder 2371 of the 540W02 volume and occupies 10 cylinders. You must ensure that these disk cylinders are available. This can be done with the **diskmap** command. *Do not simply copy this allocation.* If you do not use CMS (under the ZOSx userids) then you do not need these minidisk.

#### 4.4.1 Running z/OS under z/VM

The process for running the full z/OS system under z/VM is as follows:

1. Log on to z/VM with user ID ZOS1 (or ZOS2).<sup>6</sup>

(If you want to use CMS, enter an **i cms** command and perform whatever CMS functions you need.)

2. Enter the commands:

```
CP TERM CONMODE 3270          (There is no prompt returned.)
CP IPL A80 LOADPARM 08A2CS    (There might be a substantial pause.)
```

If CMS is running when the TERM command is issued, it may crash. This does no harm.

3. After a while, z/VM might display a HOLDING indicator. Clear the panel, and you should see the first z/OS message. Your IPL address and LOADPARM might differ from that in the example, of course, depending on the particular needs of your z/OS operation.
4. From another z/VM 3270 terminal (on the Linux desktop or through a LAN attachment), obtain a z/VM logon panel and enter **dial zos1** in the command line at the bottom of the panel. This should give you the z/OS VTAM logo panel, and you can log on to TSO from this.

The **dial** command connects the terminal to the first unused SPECIAL device for the virtual machine named in the **dial** command. In our example, the first SPECIAL device is connected to address 701 for z/OS. z/OS VTAM owns this address and presents the VTAM logo panel when it detects a new terminal connection.

5. After logging off from TSO, the z/OS VTAM logo panel is presented. To return to the z/VM logo, it is easiest to disconnect the 3270 session and then reconnect it.

<sup>6</sup> These are the arbitrary z/VM guest names in our example. You must adjust this to whatever z/VM guest names you use.

When using the x3270 emulator, the disconnect function is a selection under the File item on the toolbar. When the disconnect is complete, a Connect item appears on the toolbar. Other 3270 emulators have various methods for disconnecting and reconnecting the sessions.

If you use the z/OS TCP/IP customization that is presented in Volume 2 of this series, you should have connectivity to the LAN and to the underlying Linux (via a tunnel).

## 4.5 zIIPs and zAAPs

A 1090 system can provide a zIIP or zAAP in place of a CP, but this reduces the number of CPs available. z/VM can provide *simulated* zIIPs or zAAPs, working only with CPs in the base 1090 system. Furthermore, z/VM can provide more logical CPs, zIIPs, and zAAPs than are present in the base 1090 system.

The definition of a z/VM guest, in the z/VM directory, can contain statements such as the following:

```
MACH ESA 5                (allow up to 5 logical processors)
CPU 0 BASE
CPU 1 TYPE zIIP
CPU 2 TYPE zAAP
```

The three logical processors (one CP, one zIIP, one zAAP) can be used even if the base 1090 definition has only a single CP (a 1090-L01, for example). Of course, there are performance implications if the number of logical processors greatly exceeds the number of “real” System z processors, but this may be acceptable for development and testing situations.

## 4.6 Paging

If you run z/OS (or another System z operating system) under z/VM on a 1090 base machine, remember that you have three levels of paging:

- ▶ The base Linux system pages whenever virtual memory usage exceeds the available real memory. Our advice to have *at least* 500 MB more real memory than your defined for System z memory is intended to minimize Linux paging. A Linux page fault in the primary 1090 CP process causes the CP to pause until the page fault is resolved.
- ▶ z/VM pages when its requirement for virtual memory exceeds the defined System z memory (which is actually base Linux virtual memory). Each z/VM guest (whether a CMS user or a whole z/OS system) resides in z/VM virtual memory. z/VM systems on larger machines, running multiple significant guests, tend to page rather heavily.
- ▶ z/OS (or another System z operating system) pages when its need for real memory (which is actually z/VM virtual memory<sup>7</sup>) exceeds whatever size was defined in the z/VM directory for the z/OS guest.

A 1090 system based on a mobile computer has limited I/O bandwidth to its disk, and that bandwidth is best used for running applications rather than for paging. Our advice is to consider your memory usage carefully when planning to use z/VM for multiple guests.

---

<sup>7</sup> While conceptually true (ignoring V=R and similar environments), the implementation details depend on the level of assists enabled through the SIE instruction.



## z/VSE notes

z/VSE™ is distributed in several formats. This brief description assumes that a file (in awstape format) is used for a simple, automatic installation. There are many options that are not explored here; the intention is to describe the most basic installation process when using zPDT.

We started with an awstape file named VSE420GA.AWS, placed in a Linux file.<sup>1</sup> We created three emulated 3390 volumes (although the third volume is unused during the installation).

```
$ alckd /z/DOSRES -d3390-3
$ alckd /z/SYSWK1 -d3390-3
$ alckd /z/SYSWK2 -d3390-3           (this volume is not required)
```

We then used **gedit** to create a devmap in /home/ibmsys1/VSE. The addresses (device numbers) in the devmap are arbitrary, but the same addresses are used for operation of the resulting z/VSE system. The addresses for z/VSE are limited to three digits.

```
[system]
memory 1600           #this size is arbitrary
3270port 3270
```

```
[manager]
name aws3274 1
device 009 3279 3274
device 060 3279 3274
device 061 3279 3274
device 062 3279 3274
device 063 3279 3274
```

```
[manager]
name awsckd 2
device 300 3390 3990 /z/DOSRES
device 301 3390 3990 /z/SYSWK1
device 302 3390 3990 /z/SYSWK2
```

<sup>1</sup> We placed the tape in /tmp/VSE420GA.AWS, although it could be placed anywhere in the Linux file systems. The file size was approximately 238 MB.

```
[manager]
name awstape 50
device 180 3480 3803 /tmp/VSE420GA.AWS #remove file name after installation
device 181 3480 3803
device 182 3480 3803
```

```
#[manager] #TCP/IP is not configured
#name awsosa 100 --path=F0 --pathtype=OSD #in our sample setup.
#device 120 osa osa --unitadd=0 #These devmap definitions could
#device 121 osa osa --unitadd=1 #be used for future TCP/IP
#device 122 osa osa --unitadd=2 #activation
#device 12F osa osad --unitadd=fa
```

```
[manager]
name awsrdr 200
device 00c 2540 2821 /home/ibmsys1/z1090/cards/*
```

```
[manager]
name awsprrt 300
device 00e 1403 2821 /home/ibmsys1/z1090/lists
```

We started 1090 operation and IPLed from the z/VSE tape:

```
$ awsstart vse --clean
$ x3270 localhost:3270 & (starts terminal at address 009)
$ x3270 localhost:3270 & (starts terminal at address 060)
ipl 180
  (wait several seconds until disk activity ceases)
  (Press Enter on the 3270 session at address 009)
DO YOU WANT TO DO AN AUTOMATIC INSTALLATION (YES/NO)
0 yes (type '0', space, and the response)
IF YOU WANT A LISTING, SPECIFY CUU OF PRINTER, ELSE (Enter)
0 (type '0' and hit Enter)
IF YOU WANT TO INSTALL VSE ON SCSI SPECIFY YES, ELSE NO
0 no
ENTER CUU OF DOSRES
0 300
ENTER CUU OF SYSWK1
0 301
START AUTOMATIC INSTALLATION (YES/NO)?
0 yes
  (Some screen output)
IF YOU WANT TO USE A 3420.....
0
  (Lots of screen output)
DO YOU WANT TO MIGRATE...(VTAMONLY)? YES/NO
0 no
DB2 IS REQUIRED FOR...DO YOU WANT TO INSTALL DB2 NOW? YES/NO
0 no (You need a second distribution tape for DB2)
DO YOU WANT TO CONFIGURE TCP/IP DURING INITIAL INSTALLATION? YES/NO
0 no
SELECT AN ENVIRONMENT OUT OF A, B, OR C
0 a
DO YOU WANT TO RUN YOUR SYSTEM WITH SECURITY ON? YES/NO
0 no
CHECK YOUR ANSWERS. DO YOU WANT TO CONTINUE? YES/NO
```

```

0 yes
IS THE LOCAL CONTROL UNIT AN SNA CU? YES/NO
0 no
ENTER A 3270 ADDRESS (CUU) OR "END"
0 060
ENTER A 3270 ADDRESS (CUU) OR "END"
0 061
ENTER A 3270 ADDRESS (CUU) OR "END"
0 062
CHECK YOUR ANSWERS... CONTINUE? YES/NO
0 yes
  (Screen output)
// PAUSE
0
  (Lots of screen output, pauses, several minutes work)
  (Logo should appear on the second 3270 session)
  (Last message is about LSR pool1

```

At this point, move to the second 3270 session (at address 060) and continue with the initial installation process. Do not disrupt the installation at this point because the following logon (as userid POST) can be done only once.

Logon as user POST with password BASE

```

  Convention.....2
(Enter)

```

Complete some of the customer information

```

  Customer name _____
  Address _____
  Phone number _____
  Programmer name _____
(Enter)

```

This should produce a new logo panel on the same (060) terminal.

Logon as user SYSA with password SYSA

Set a new password for user SYSA

```

  New password _____
  New password _____
  Old password SYSA _____
(Enter)

```

```

          /VSE FUNCTION SELECTION
1 Installation
2 Resource Definition
3 Operation
4 Problem Handling
5 Program Development
6 Command Mode
7 CICS-Supplied Transactions

```

Navigate to **2** (Resource Definition) → **4** (Hardware Configuration and IPL) → **1** (Configure Hardware).

OPT	ADDR	DEVICE	DEVICE TYPE
—	009	?	3277
—	060	?	3277
—	061	?	3277
—	062	?	3277
—	063	?	3277

Enter **1** in the OPT column for ADDR 009.

Select **1** (Local non-SNA) in the next screen.

Select **43x80** (or whatever you want) in the SEL column of the next screen.

Repeat the last three steps for addresses 060, 061, 062, 063.

**F5** - Process

**F3** - End

This completes the basic z/VSE installation process. Userid SYSA is the administrative userid, with the password you selected earlier. We did note that the Configure Hardware panel said our two disks (DOSRES and SYSWK1) were 3390-9 units, whereas they were really 3390-3 units; this seemed to have no ill effects.

The z/VSE system can be shut down as follows (working on the 009 terminal session):

```
MSG F2,DATA='CEMT P SHUT I'  
z net,quick  
pend
```

More information about z/VSE installation can be found in *IBM z/VSE Installation*, SC33-8302.



## Multiple zPDT instances

zPDT supports both guest operations (under z/VM) and multiple instances of zPDT. Both are ways to run multiple z/OS or other operating systems.

See additional notes elsewhere about multiple instances using LANs (in “LAN notes” on page 113) and using cryptographic adapter functions (in “Cryptographic adapter” on page 141).

## 6.1 Multiple instances or guests

We strongly suggest you use a single 1090 instance, as described in this series of documents, to become familiar with basic 1090 operation. Also, remember that you cannot exceed three System z CPs (with a 1090-L03) regardless of how the CPs are distributed across instances and you cannot exceed the number of processors in your Linux system in a single instance.

Multiple *instances* means running more than one copy of zPDT. Each instance must run under a different Linux userid. This can be accomplished by logins through telnet (or ssh) or by careful use of `su` commands from different windows on the Linux desktop. The `.bashrc` file of each userid must have the appropriate export statements. Each instance must have its own devmap.

The use of TCP/IP interfaces is an essential part of this discussion. For this reason we combine a discussion of multiple zPDT instances with the use of guests under z/VM in a single instance. Our examples use OSD (QDIO) interfaces for TCP/IP. It is possible to use OSE interfaces (non-QDIO) for TCP/IP; however, in the case of multiple instances using OSE (through a single emulated OSA) you *must* configure the OSA Address Table (OAT) using the OSA/SF utility.

## 6.2 Multiple guests in one instance

A typical z/VM configuration is outlined in Figure 6-1. z/VM itself typically “owns” all the 3270 sessions. Guests (z/OS, CMS) acquire a 3270 when a user logs on to the guest or uses a `dial` command.

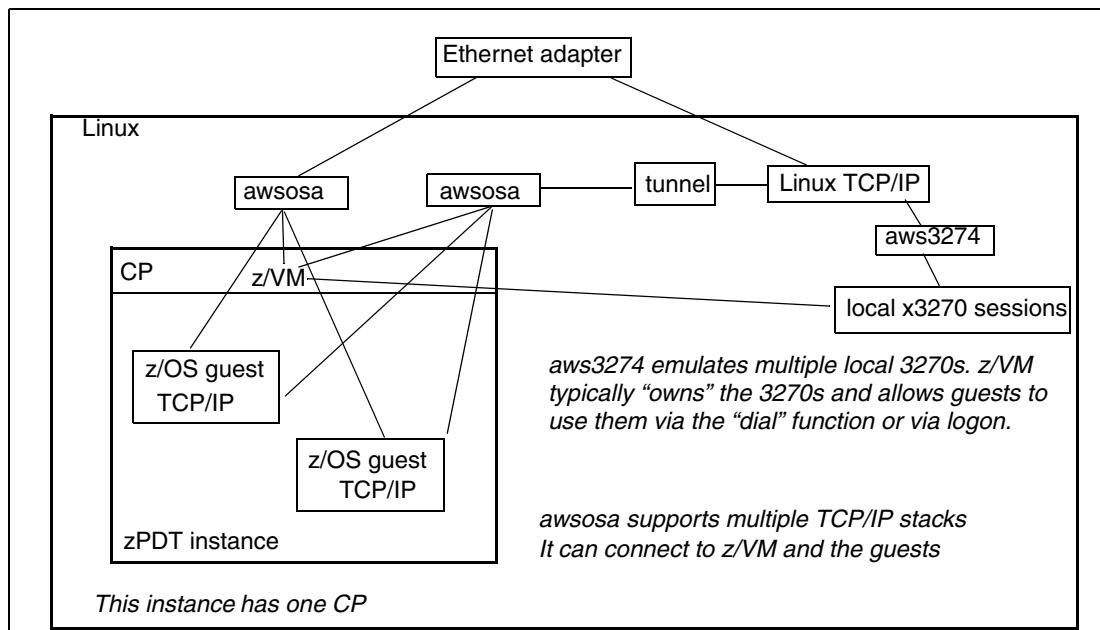


Figure 6-1 Guests in a single zPDT instance

Each guest (under z/VM) can access a LAN interface. The `awsosa` device manager can handle up to 16 of these “stacks”. An `awsosa` device manager using a tunnel to Linux can be used, as shown in the illustration. Each z/VM guest would use different IP addresses on each OSA interface. Alternatively, not shown in the illustration, z/VM could establish an internal

VSWITCH for guest use. Using the IP address patterns from our other examples, we might have the following addresses in Figure 6-1:

192.168.0.60	Linux IP address for the Ethernet adapter
192.168.0.60 port 3270	address for external TN3270e connections to aws3274
10.1.1.1	Linux IP address for the tunnel interface
192.168.0.61	z/VM IP address for Ethernet
10.1.1.2	z/VM IP address for the tunnel interface
192.168.0.62	z/OS #1 address for Ethernet
10.1.1.3	z/OS #1 address for the tunnel interface
192.168.0.63	z/OS #2 address for Ethernet
10.1.1.4	z/OS #2 address for the tunnel interface
127.0.0.1	localhost connection for local x3270 sessions

### 6.3 Independent instances

We can have two independent instances, meaning that emulated I/O devices are not shared between the instances. In common terms, there is no shared DASD (or any other shared device).

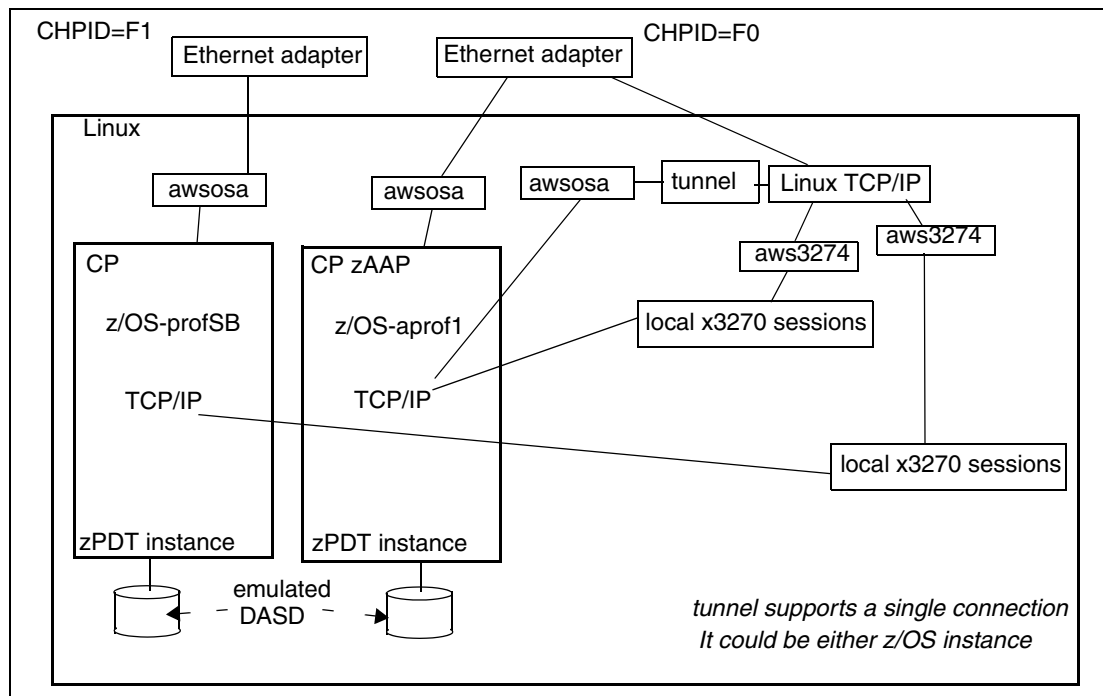


Figure 6-2 Independent instances

We assume an L03 license (and a base Linux machine with two or more processors) and we have assigned two CPs (a CP and a zAAP) to one instance and one CP to the other instance. Notice that different port numbers are needed in the 3270port statements in the devmaps. Emulated device addresses (device numbers) are independent between the instances and both might use the same addresses, as described here.

Each emulated OSA requires its own Ethernet adapter and two adapters are needed in this case. Two emulated OSAs cannot share an Ethernet adapter. We arbitrarily assigned a tunnel interface to one of the instances.<sup>1</sup> This example suggests LCS (non-QDIO) mode for both instances, but they could both be QDIO or a mixture of LCS and QDIO.<sup>2</sup>

Simplified devmaps, matching Figure 6-2, might be as follows:

```
(file /home/ibmsys1/aprof1)
[system]
memory 800m                # emulated zSeries to have 800 MB memory
3270port 3270              # tn3270e connections specify this port
processors 2 cp zaap        # one CP and one zAAP

[manager]
name awsckd 0001           # define a single 3390 disk
device 0a80 3390 3990 /z/SARES1

[manager]
name aws3274 0003          # define two local 3270s
device 0700 3279 3274 mstcon
device 0701 3279 3274 tso

[manager]
name awsosa 00C0 --path=F0 --pathtype=OSE
device E20 osa osa --unitadd=0
device E21 osa osa --unitadd=1

[manager]
name awsosa 00A0 --path A0 --pathtype=OSE --tunnel_intf=y
device E22 osa osa --unitadd=0
device E23 osa osa --unitadd=1
```

```
(file /home/ibmsys2/profSB)
[system]
memory 1000m               # emulated zSeries to have 1GB memory
3270port 3271             # tn3270e connections specify this port
processors 1

[manager]
name awsckd 0001           # define a single 3390 disk
device 0a80 3390 3990 /z/SA9999
device 0200 3390 3990 /z/VMBASE

[manager]
name aws3274 0003          # define two local 3270s
device 0700 3279 3274 L700
device 0701 3279 3274 L701

[manager]
name awsosa 0123 --path=F1 --pathtype=OSE
device E20 osa osa --unitadd=0
device E21 osa osa --unitadd=1
```

Startup for these instances, working from the Linux desktop, might go as follows:

```
(login as root; open a terminal window)
# xhost +                                allow multiple users to start x3270
```

<sup>1</sup> Starting with the 1090 release available in February 2010, multiple tunnel interfaces may be defined.

<sup>2</sup> Since each instance has its own OSA, the user is not required to do OAT configuration if he uses the default OAT definitions.

```

# su ibmsys1
$ cd /home/ibmsys1
$ awsstart aprof1                                working as ibmsys1
  (startup messages)                            ibmsys1 instance
$ x3270 -port 3270 mstcon:localhost &            working as ibmsys1
$ x3270 -port 3270 tso:localhost &              working as ibmsys1
$ ipl a80 parm 0a8200                             working as ibmsys1, IPL z/OS
(open another terminal window)
# su ibmsys2
$ cd /home/ibmsys2
$ awsstart profSB                                working as ibmsys2
  (startup messages)                            ibmsys2 instance
$ x3270 -port 3271 localhost &                  working as ibmsys2
$ x3270 -port 3271 localhost &                  working as ibmsys2
$ ipl 200                                         working as ibmsys2, IPL VM

```

Each instance is started with its own devmap. Each devmap must specify a different port address for local 3270 connections. Each instance must specify different emulated disk volumes. Attempting to share an emulated disk volume in this situation (by specifying the same Linux file for the emulated volume) will result in corrupted data on the volume.

The use of **xhost** + presents a security exposure; you should tailor this command to suit your security environment

## 6.4 Instances with shared I/O

It is possible for multiple instances to share certain devices, such as emulated DASD and emulated OSA. Also, a single pool of 3270 devices can be used and accessed via a common Linux port number, although this option has more complex side effects. The most common use of a shared configuration is to provide *shared DASD* among the instances.

Please note that zPDT does not support the VMAC function of z/OS. The only virtual mac supported is generated on z/VM with the layer-2 vswitch.

A configuration with shared I/O devices requires a *group controller*; see Figure 6-3. The group controller is similar to another zPDT instance, but without an associated CP or memory. The group controller must have its own Linux userid, its own devmap, and be started with its own **awsstart** command. It must be started before other instances are started. As a basic concept, the I/O devices defined in the group controller's devmap are inherited and shared by the other instances.

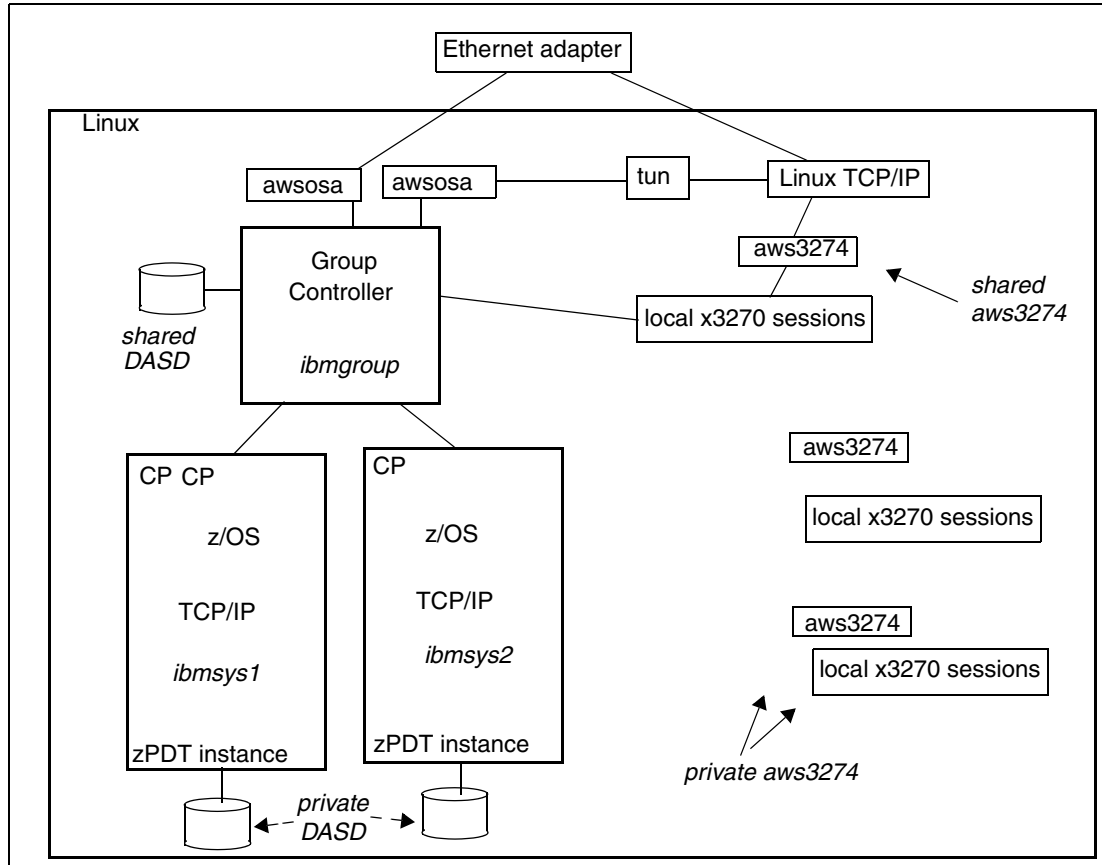


Figure 6-3 Shared emulated I/O

The Linux userid associated with the group controller must have the proper path information set in the `.bashrc` file, just like that for the userids associated with each instance. All the userids involved (the group controller and the instances) must be in the same Linux group; this is group `ibmsys` in our examples. All the emulated volume files must be readable and writable (possibly via the `groupid`) by all the userids involved.<sup>3</sup> For our example, assume we have three userids defined (`ibmgroup`, `ibmsys1`, and `ibmsys2`) and all are in group `ibmsys`. We could define three `devmaps`, as follows:

```

/home/ibmgroup/group1
[system]
members ibmsys1 ibmsys2          # userids for the instances

[manager]
name awsckd 8765
device A80 3390 3990 /z/Z9RES1
device A81 3390 3990 /z/Z9RES2
device A82 3390 3990 /z/Z9SYS1
device A83 3390 3990 /z/Z9RES3
device A84 3390 3990 /z/Z9USS1
device A90 3390 3990 /z/SARES1

[manager]
name awsosa 1223 path=F0 --pathtype=OSD

```

<sup>3</sup> This is controlled by normal Linux permission settings for each file. For example, the command `chmod g+w /z/*` could be used to make all the files in directory `/z` writable by members of the current group.

```

device 400 osa osa --unitadd=0
device 401 osa osa --unitadd=1
device 402 osa osa --unitadd=2
device 403 osa osa --unitadd=3
device 404 osa osa --unitadd=4
device 405 osa osa --unitadd=5
device 406 osa osa --unitadd=6
device 407 osa osa --unitadd=7

```

*/home/ibmsys1/aprof1*

```

[system]
memory 800m
3270port 3270
processors 1
group ibmgroup #userid of the group controller

```

```

[manager]
name aws3274 4455
device 0700 3279 3274 mstcon
device 0701 3279 3274

```

*/home/ibmsys2/aprofSB*

```

[system]
memory 1000m
3270port 3271
processors 2
group ibmgroup #userid of the group controller

```

```

[manager]
name aws3274 5544
device 0700 3279 3274 mstcon
device 0701 3279 3274

```

Notice the two new devmap statements in this example. Both are in the [system] stanzas:

- ▶ **members name1 name2** is used in the group controller definitions and specifies the Linux userid associated with each instance in the group.
- ▶ **group cnt1name** is used in each instance and specifies the Linux userid associated with the group controller.

TN3270e sessions are directed to the desired instance by using the appropriate 3270port number:

```

$ x3270 -port 3270 localhost &           connects to the ibmsys1 instance
$ x3270 -port 3271 localhost &           connects to the ibmsys2 instance

```

There is no need to coordinate device numbers or unit addresses among multiple instances using shared OSA. For example, each instance might use an OSA interface at addresses 400-403. Each instance might start unit addresses (specified in the devmap) at address zero. (Multiple guests under z/VM, in a single instance, must manage the addresses properly. Do not confuse multiple guests under z/VM with multiple instances.)

Only DASD (CKD or FBA) and OSA can be shared. Additional devices, such as tape drives, can be included in the group controller devmap. These additional device definitions are inherited by all instances, but each instance uses the definitions as though they were part of the devmap for that instance. Notice that the two instances in the previous example have

different 3270port addresses; we elected to not use shared 3270 definitions in this example.<sup>4</sup> No DASD is defined for the instances in this example; the instances will share the DASD defined for the group controller.

Note that all sharing instances use the same addresses (device numbers) for the shared devices. There is no provision to have different addresses (for different instances) for the same shared device.

If a zPDT instance operates under the group controller, then any OSA devices might be shared devices, managed by the group controller, or each instance may have a private OSA.<sup>5</sup> If the OSA is used in OSE (non-QDIO mode) then the OAT definitions must be customized with the names of the instance members (specified as “MEMBER names”) and the IP address(es) for each instance.

Standard operating rules still apply, of course. We cannot IPL the same z/OS system into two instances at the same time.<sup>6</sup> In our small example, we have two z/OS systems (the second one is on the SARES1 volume provided with the AD-CD package). In the absence of shared ENQ functions<sup>7</sup>, you must manage any active data set sharing. The 1090 correctly emulates disk RESERVE and RELEASE functions and these protect VTOC and catalog updates in the normal z/OS manner.

Startup for our controller and two instances, working from the Linux desktop, might go as follows:

```
(login as root; open a terminal window)
# xhost +                allow multiple users to start x3270
# su ibmgroup            work as group controller
$ cd /home/ibmgroup
$ awsstart group1      start group controller
    (startup messages)
(open another terminal window)
# su ibmsys1
$ cd /home/ibmsys1
$ awsstart aprof1     working as ibmsys1
    (startup messages)   ibmsys1 instance
$ x3270 -port 3270 mstcon@localhost &  working as ibmsys1
$ x3270 -port 3270 tso@localhost &    working as ibmsys1
$ ipl a80 parm 0a8200  working as ibmsys1, IPL z/OS
(open another terminal window)
# su ibmsys2
$ cd /home/ibmsys2
$ awsstart profSB     working as ibmsys2
    (startup messages)   ibmsys2 instance
$ x3270 -port 3271 mstcon@localhost &  working as ibmsys2
$ x3270 -port 3271 tso@localhost &    working as ibmsys2
$ ipl a90 parm 0a90sa  working as ibmsys2, IPL z/OS
```

<sup>4</sup> An example using a single 3270 port number is given later in the text.

<sup>5</sup> zPDT releases prior to 39.16 could not have multiple OSA tunnel devices; this limitation was removed with release 39.16 and later releases.

<sup>6</sup> This statement ignores situations where the use of different PARMLIB members allows the same z/OS to be IPLed in multiple LPARs or instances. This involves separate paging, spooling, and various VSAM data sets for each LPAR/instance. The z/OS AD system used for many of our examples is not set up for this type of usage.

<sup>7</sup> Sharing ENQ/DEQ functions is typically done the GRS functions of a sysplex configuration. We do not have a sysplex here and there are no global ENQ/DEQ controls.

In this example, we used a different terminal window to start the group controller and each z/OS instance. This allows us to send commands (such as `awsstop`) to the appropriate application later.

## 6.5 Additional shared functions

The previous section outlines the key shared device usage rules, as they apply to DASD and OSA devices. The group controller can also include a shared `aws3270` function and passive definitions that are inherited by all instances.

### Shared `aws3270` options

The group controller devmap can include `aws3270` definitions such as the following:

```
/home/ibmgroup/group1
[system]
3270port 3270
members ibmsys1 ibmsys2

[manager]
name aws3270 1234
device 700 3279 3274 mstcon
device 701 3279 3274
device 702 3270 3274
```

In this case the group controller has specified a port address for TN3270e connections. Each instance inherits the complete set of 3270 device definitions (700, 701, 702) but not the 3270port address. That is, each instance has a 3270 at address 700, 701, and so forth. If a user starts a TN3270e session connected to the 3270port number on Linux, he has several options:

```
$ x3270 -port 3270 localhost &           Example 1
$ x3270 -port 3270 ibmsys1@localhost &   Example 2
$ x3270 -port 3270 ibmsys2.701@localhost & Example 3
$ x3270 -port 3270 ibmsys1.mstcon@localhost & Example 4
```

In Example 1 the desired instance is not specified. In this case, the group controller will display a selection menu (on the new 3270 session) and you must indicate which instance you want and, optionally, which terminal in that instance.<sup>8</sup> This selection menu is illustrated in Figure 6-4. In Example 2, the first available 3270 in the `ibmsys1` instance is assigned. (The *instance name* corresponds to the userid that started the instance.) Examples 3 and 4 specify both an instance name and the 3270 device identifier.

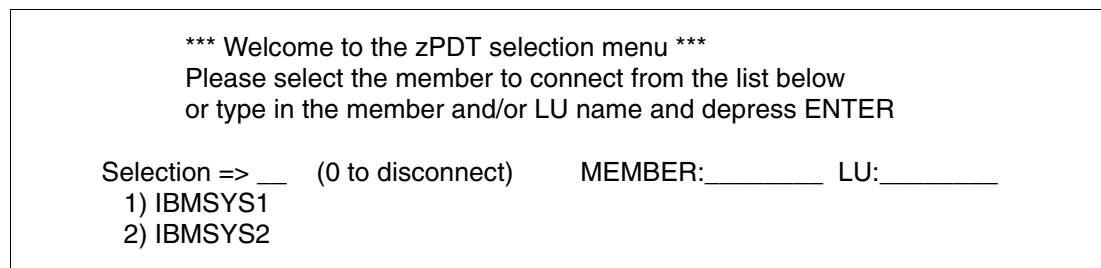


Figure 6-4 Selection menu with two instances running

<sup>8</sup> If a specific terminal within the instance is not specified (by address, or by LU name) then the first available 3270 in that instance is used.

As a reminder, you can specify a different 3270port number and aws3274 device definitions in each instance. In this case the shared aws3274 conditions do not apply. You could specify a 3270port number and aws3274 devices in the group controller and also specify aws3274 devices in each instance (but without a 3270port number in the instances). In this case all the 3270 devices (from the controller list that is inherited by all instances, and from the unique list in each instance) can be accessed from the selection menu.

Yet another option exists for accessing shared aws3274 functions. This involves an inetd service that automatically detects which 1090 instances are running and constructs a selection menu based on this information. The inetd setup varies with different Linux distributions.

### **Inherited devices**

The group controller devmap can include definitions for device managers other than awsscd, awsfba, awsosa, and aws3270. For example, it might contain the following:

```
/home/ibmgroup/group1
[system]
membersibmsys1 ibmsys2

[manager]
name awstape 4444
device 580 3480 3480
device 581 3480 3480
```

In this case each instance (ibmsys1 and ibmsys2) will have emulated 3480 tape drives at addresses 580 and 581. There is no connection between these drives in the two instances. It is exactly as though the awstape stanza appeared in the devmaps for each instance. The sole purpose is to remove the necessity for defining these devices for each instance. This is not very meaningful for a small device list as shown here, but might be more meaningful for longer lists.



## Using awscmd

The awscmd device manager provides a “device” that appears to System z software as a tape drive. Its function is to send commands (and data) to the underlying Linux and then receive the output from the Linux command. Any Linux command may be sent, including those that could destroy the Linux system. Obviously, this device manager should be used with care and may not be appropriate for a zPDT environment that can be accessed by untrusted users.

Configuration is similar to other device managers:

```
[manager]
name awscmd 20
device 580 3480 3480
device 581 3480 3480
```

The CUNUMBR (which is 20 in this example) is an arbitrary hexadecimal number (up to four digits) that cannot duplicate the CUNUMBR used with any other device manager. We show two devices here, but typically only one is needed. The device type can be 3420, 3422, 3480, 3490, or 3590; these are the tape device types emulated by zPDT. The device number (580, 581) should match a corresponding device type in your z/OS IODF. (Any device number may be used with z/VM.)

The intended operation is as follows:

1. A rewind is issued to the device.
2. The desired Linux command (expressed in EBCDIC) is written to the device.
3. Any stdin data to be used by the Linux command is written to the device.
4. EBCDIC to ASCII translation is done automatically.
5. A tape mark is written to the device.
6. At this point, the awscmd device manager submits the command (and data) to Linux through a shell that does not appear on the Linux window. The current Linux directory for the command is the directory that was used to start zPDT.
7. When the awscmd function completes there are four files on the pseudo-tape device:
  - The command file that was submitted to Linux (with redirection operands that were automatically added by awscmd)

- The stdout data from the Linux command
  - The stderr data from the Linux command
  - The return code (converted to characters) from the Linux command
8. The output (on the pseudo-tape) has been converted to EBCDIC.
  9. Two tape marks are at the end of the pseudo-tape.

## Restrictions

The command you send to Linux cannot include any redirection (< or > characters), asynchronous indicator (& character), or pipe (“|” or vertical bar character). The pseudo-tape device will appear to be busy while Linux is executing the command. Any Linux command that creates substantial delays (of many seconds) may cause I/O timeout errors to be generated in z/OS.

At the time of writing, some characters did not survive the EBCDIC to ASCII conversion when included in SYSIN data. These were the tilde(~), caret (^), colon(:), double quote(“), less-than(<), greater-than(>), and question mark(?). This restriction may change in later versions of awscmd.

## 7.1 Sample z/VM script

The following REXX script assumes that the awscmd device is attached to the CMS user as device 28F:

```

/* CMS REXX script to execute a Linux command */
/* */
/* format: */
/*   oscmd Linux-command (tape-address) */
/* */
/* The tape-address is optional; defaults to 28F */
/* */
Trace off;
Parse arg cmd '(' tDev;
if (length(tDev) = 0)
  then tDev = 28F;
/* Write the Linux command string to the tape */
"tape rew (" tDev;
"pipe var cmd | tape" tDev;
"tape wtm (" eDev;
/* Read the stdout file from the tape */
"tape rew (" tDev;
"tape fsf (" tDev; /* skip over input file */
say "STDOUT output-----"
"pipe tape" tDev "| console";
/* Read the stderr file from the tape */
"say "STDERR output -----"
"pipe tape" tDev "| console"
/* Read the return code from the tape */
"pipe tape" tDev "| console"
/* end this script */
return(0);

```

This script could be used from z/VM as follows:

```
att 280 * 28f          (attach the awscmd pseudo device)
```

```

TAPE 0280 ATTACHED TO ZVMTEST 028F
Ready;
oscmd ls -al
STDOUT output-----
total 21699469
drwxrwxr-x 2 zvmtest zvmtest      4096 Aug  7 20:51 .
drwdr-xr-x 8 zvmtest zvmtest      4096 Aug  7 20:37 ..
-rw-rw-r-- 1 zvmtest zvmtest 2846431232 Jul  8 09:58 5300PT.ckd
-rw-rw-r-- 1 zvmtest zvmtest 2846431232 Jul  8 10:08 530PAG.ckd
  (etc to list all the entries in the current Linux directory)
STDERR output----
COMMAND return code ----
0
Ready;

```

## 7.2 z/OS usage

Using the awscmd device with z/OS is more challenging than using it with z/VM for several reasons:

- ▶ Tape drives are not readily manipulated by TSO users.
- ▶ z/OS wants to check tape volumes for labels, even if you specify a no-label tape volume.
- ▶ For practical purposes, a batch program must be written to utilize the awscmd functions.

You can write your own program. You may want to examine the sample batch program we have provided, listed in “Sample z/OS program for awscmd” on page 84. This program looks for a PARM field on the EXEC JCL statement and sends this as the command to Linux. If no PARM field is present, it opens DDname SYSIN and sends the first line as the Linux command and sends any additional lines as stdin for the Linux command. Output from the awscmd function is printed on DDNAME SYSPRINT. A JCL DD statement is needed to allocate the pseudo-tape drive for awscmd. Our example uses address 580 for the pseudo-tape because this is a known 3480 address for our z/OS system.

Our devmap contains:

```

[manager]
name awscmd 20
device 580 3480 3480

```

Our sample program requires that an MVS initiator be enabled for BLP processing. This can be done by changing the JES2 startup parameters or (for the duration of an IPL) by entering the following command on the MVS console:

```
$T JOBCLASS(A),BLP=YES           (you might want to use jobclass other than A)
```

We then mount a tape on the pseudo-tape drive for continued use. This avoids having to respond to a mount message every time the sample program is run. The MVS command:

```
MOUNT 580,VOL=(NL,123456)
```

followed by the zPDT command:

```
$ ready 580
```

provides the necessary setup.

An example of using the sample program might be:

```
//OGDEN22 JOB 1,OGDEN,MSGCLASS=X,MSGLEVEL=(0,0)
// EXEC PGM=AWSCMDX,PARM='ls -al '
//STEPLIB DD DSN=OGDEN.LIB.LOAD,DISP=SHR
//TAPE DD UNIT=(580,,DEFER),VOL=SER=123456,LABEL=(1,BLP),DSN=X
//SYSPRINT DD SYSOUT=*
```

The output (viewed from JES2 spool using SDSF) contains the usual JES2 messages and a SYSOUT data set like the following:

```
COMMAND:  ls -al 1>/tmp/AWSCMD-xxx-out.txt 2>/tmp/etc.xxetc
</tmp/AWSCMD.xxetc
STDOUT: total 21699469
STDOUT: drwxrwxr-x 2 ibmsys1 ibmsys1      4096 Aug  1 20:01 .
STDOUT: drwdr-xr-x 4 ibmsys1 ibmsys1      4096 Aug  1 20:02 ..
STDOUT: -rw-rw-r-- 1 ibmsys1 ibmsys1 2846431232 Aug  8 09:58 WORK01
STDOUT: -rw-rw-r-- 1 ibmsys1 ibmsys1 2846431232 Aug  8 10:08 WORK02
      (etc to list all the entries in the current Linux directory)
STDERR:
RTNCDE: 0
```

The Linux command that is actually executed contains redirection operators that are automatically added by awscmd. You can see these operators in the output listing; they are only suggested in the sample output shown here.

A second example, using SYSIN data to create a new Linux file, could be:

```
//OGDEN22 JOB 1,OGDEN,MSGCLASS=X,MSGLEVEL=(0,0)
// EXEC PGM=AWSCMDX
//STEPLIB DD DSN=OGDEN.LIB.LOAD,DISP=SHR
//TAPE DD UNIT=(580,,DEFER),VOL=SER=123456,LABEL=(1,BLP),DSN=X
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
tee my.new.file
This is a line of data for the new file
This is another line of data
And yet another
/*
```

This example would create Linux file *my.new.file* (in the Linux directory used to start zPDT) with the indicated lines in the file. A fully-qualified Linux file name could be used with either of the examples. The output for the second example would list all the data lines with the COMMAND output and then list them again for STDOUT output. (This is because the **tee** command writes all the stdin data to stdout. The **tee** command appends the new data if the specified file already exists.)

## 7.2.1 Sample z/OS program for awscmd

The following listing is for a z/OS batch program that exploits the awscmd command processor to send a command to the underlying Linux and receive the results.

This sample program is simple-minded in many respects and is not intended to illustrate the best programming techniques, but it should be fairly readable. Errors result in WTO messages; this is a poor design for significant applications but it should be reasonable for many 1090 environments. The program includes a two-second wait before reading the results of the Linux command. This wait is not necessary and can be removed. (The pseudo-tape

device remains busy with the preceding WTM operation until the Linux command completes.)

The TSO test environment options (in the compile and link steps) are not required.

```
//OGDEN90 JOB 1,OGDEN,MSGCLASS=X
// EXEC ASMACLG,PARM.C='NOXREF,TEST',PARM.L='NOLIST,NOMAP,TEST'
//*      PARM.G='ls -al '
//C.SYSIN DD *
        PRINT NOGEN
*
* SEND COMMAND TO LINUX VIA AWSCMD, READ THE RESULT
* JES2 MUST ALLOW BLP FOR THE JOB CLASS THAT IS USED
*
* $T JOBCLASS(A),BLP=YES CAN BE USED FOR TEMPORARY CHANGES
*
* MOUNT 580,VOL=(NL,123456)
* awsmount 580 -o /z/123456
*
*
*//TAPE DD UNIT=(580,,DEFER),LABEL=(1,BLP),VOL=SER=XXXXXX,DSN=X
*//SYSPRINT DD SYSOUT=*      (OUTPUT FROM LINUX)
*
AWSCMDX CSECT
        STM 14,12,12(13)  SAVE CALLER'S REGISTERS
        LR 12,15          USE ENTRY-POINT BASE REGISTER
        USING AWSCMDX,12
        ST 13,SAVEAREA+4  CALLER'S SAVEAREA ADDRESS
        LA 2,SAVEAREA     MY SAVEAREA ADDRESS
        ST 2,8(13)       STORE A(MY SAVE AREA) IN CALLER
        LR 13,2          MY SAVEAREA IN R13
        LR 11,12         SECOND BASE REGISTER
        A 11,=F'4096'
        USING AWSCMDX+4096,11 NOT REALLY NEEDED
* CHECK PARM DATA
        L 1,0(1)         GET ADDRESS OF PARM FIELD
        LH 2,0(1)       GET LENGTH OF PARM FIELD
        LTR 2,2         CHECK IT
        BZ NOPARM       BRANCH IF LENGTH = ZERO
        BCTR 2,0        SUBTRACT 1 FROM LENGTH
        EX 2,MOVPARM    MOVE PARM TO MY WORK AREA
        B A            GO USE PARM DATA
* TRY SYSIN FOR THE INPUT DATA
NOPARM OPEN (SYSIN,(INPUT))
        TM SYSIN+48,X'10' DID OPEN WORK?
        BZ NOINPUT     IF NOT, BRANCH
        MVI PARMFLG,X'FF' REMEMBER TO USE SYSIN
* OPEN PSEUDO-TAPE DEVICE AND SYSPRINT
A OPEN STAPEO ASSUME BLP ON DD STATEMENT
        TM STAPEO+48,X'10' DID OPEN WORK?
        BZ ERR10      IF NOT, BRANCH
A10 OPEN (PRINT,(OUTPUT))
        TM PRINT+48,X'10' DID OPEN WORK?
        BZ ERR11
* REWIND THE PSEUDO TAPE
B MVI SECB,X'00' ZERO MY ECB
```

```

        LA    1,SCREW          ADDRESS of CCW(S)
        ST    1,SIOB+16       PLACE IN IOB
        EXCP  SIOB             REWIND TAPE
        WAIT  ECB=SECB        WAIT FOR IT
        TM    SECB,X'20'      COMPLETED OK?
        BZ    ERR1            IF NOT, BRANCH
* PREPARE TO WRITE THE COMMAND RECORD
C        CLI  PARMFLG,X'00'   USING PARM DATA?
        BE    C2              IF SO, BRANCH
C1       GET  SYSIN,BUFFER    READ RECORD FROM SYSIN
* BACKSCAN TO REMOVE TRAILING BLANKS
C2       LA    3,BUFFER+99    MAX 100 BYTES
C3       CLI  0(3),C' '      A BLANK?
        BNE  C4              IF NOT, BRANCH
        S    3,=F'1'         BACK UP 1 IN BUFFER
        B    C3              LOOP UNTIL NON-BLANK FOUND
C4       LA    4,BUFFER      A(BEGINNING OF BUFFER)
        LA    3,1(3)         ADJUST FOR LAST CHARACTER
        SR    3,4            A(LAST NON-BLANK) - A(BEGINNING)
        BP    C5            IF POSITIVE, BRANCH
        LA    3,1           IF NOT, MAKE LENGTH = 1 BYTE
C5       STH  3,SCCWRITE+6   CHANGE CCW LENGTH FIELD
        MVI  SECB,X'00'     ECB
        LA    1,SCCWRITE    CCW
        ST    1,SIOB+16     IN IOB
        EXCP  SIOB          WRITE RECORD
        WAIT  ECB=SECB      WAIT
        TM    SECB,X'20'   OK?
        BZ    ERR2         IF NOT, BRANCH
        CLI  PARMFLG,X'00' USING PARM DATA?
        BE    D            IF SO, BRANCH
        MVC  BUFFER(132),BLANKS
        B    C1           LOOP TO GET ALL SYSIN DATA
* WRITE A TAPE MARK
D        MVI  SECB,X'00'     ECB
        LA    1,SCCWTM      CCW
        ST    1,SIOB+16     IN IOB
        EXCP  SIOB          WRITE TAPE MARK
        WAIT  ECB=SECB      WAIT
        TM    SECB,X'20'   OK?
        BZ    ERR3         IF NOT, BRANCH
        B    EE
*
* WAIT AN ARBITRARY TWO SECONDS (CAN BE REMOVED)
*
EE       STIMER WAIT,BINTVL=SEC2
*
* REWIND THE TAPE
E        MVI  SECB,X'00'     ZERO ECB
        LA    1,SCREW        CCW
        ST    1,SIOB+16     IN IOB
        EXCP  SIOB          REWIND TAPE
        WAIT  ECB=SECB      WAIT
        TM    SECB,X'20'   OK?
        BZ    ERR4         IF NOT, BRANCH

```

```

* READ FIRST FILE
F      MVC  BUFFER2(80),BLANKS
      MVC  BUFFER2+0080(80),BLANKS
      MVC  BUFFER2+0160(80),BLANKS
      MVI  SECB,X'00'      ZERO THE ECB
      LA   1,SCCREAD      ADDRESS of CCW(S)
      ST   1,SIOB+16      PLACE IN IOB
      EXCP SIOB           READ
      WAIT ECB=SECB       WAIT FOR IT
      TM   SECB,X'20'     COMPLETED OK?
      BO   F1             IF YES, BRANCH
      TM   SIOB+12,X'01' TAPE MARK?
      BO   G              IF YES, BRANCH
      B    ERR5
F1     MVC  BUFFER(132),BLANKS
      MVC  BUFFER(09),=C'COMMAND: '
      MVC  BUFFER+09(132),BUFFER2
      PUT  PRINT,BUFFER
      B    F              LOOP UNTIL ALL RECORDS READ

* READ SECOND FILE
G      MVC  BUFFER2(80),BLANKS
      MVC  BUFFER2+0080(80),BLANKS
      MVC  BUFFER2+0160(80),BLANKS
      MVI  SECB,X'00'      ZERO THE ECB
      LA   1,SCCREAD      ADDRESS of CCW(S)
      ST   1,SIOB+16      PLACE IN IOB
      EXCP SIOB           READ
      WAIT ECB=SECB       WAIT FOR IT
      TM   SECB,X'20'     COMPLETED OK?
      BO   G1            IF YES, BRANCH
      TM   SIOB+12,X'01' TAPE MARK?
      BO   H              IF YES, BRANCH
      B    ERR5
G1     MVC  BUFFER(132),BLANKS
      MVC  BUFFER(09),=C'STDOUT: '
      MVC  BUFFER+09(132),BUFFER2
      PUT  PRINT,BUFFER
      B    G              LOOP UNTIL ALL RECORDS READ

* READ THIRD FILE
H      MVC  BUFFER2(80),BLANKS
      MVC  BUFFER2+0080(80),BLANKS
      MVC  BUFFER2+0160(80),BLANKS
      MVI  SECB,X'00'      ZERO THE ECB
      LA   1,SCCREAD      ADDRESS of CCW(S)
      ST   1,SIOB+16      PLACE IN IOB
      EXCP SIOB           READ
      WAIT ECB=SECB       WAIT FOR IT
      TM   SECB,X'20'     COMPLETED OK?
      BO   H1            IF YES, BRANCH
      TM   SIOB+12,X'01' TAPE MARK?
      BO   J              IF YES, BRANCH
      B    ERR5
H1     MVC  BUFFER(132),BLANKS
      MVC  BUFFER(09),=C'STDERR: '
      MVC  BUFFER+09(132),BUFFER2

```

```

        PUT PRINT,BUFFER
        B H LOOP UNTIL ALL RECORDS READ
* READ FOURTH FILE
J MVC BUFFER2(80),BLANKS
  MVC BUFFER2+0080(80),BLANKS
  MVC BUFFER2+0160(80),BLANKS
  MVI SECB,X'00' ZERO THE ECB
  LA 1,SCCREAD ADDRESS of CCW(S)
  ST 1,SIOB+16 PLACE IN IOB
  EXCP SIOB READ
  WAIT ECB=SECB WAIT FOR IT
  TM SECB,X'20' COMPLETED OK?
  BO J1 IF YES, BRANCH
  TM SIOB+12,X'01' TAPE MARK?
  BO K IF YES, BRANCH
  B ERR5
J1 MVC BUFFER(132),BLANKS
   MVC BUFFER(09),=C'RTNCDE: '
   MVC BUFFER+09(132),BUFFER2
   PUT PRINT,BUFFER
   B J LOOP UNTIL ALL RECORDS READ
*
K SR 1,1 NOP
*
CLOSE CLOSE (STAPE0,,PRINT)
RETURN L 13,4(13) GET A(CALLER'S SAVE AREA)
      LM 14,12,12(13) RESTORE CALLER'S REGISTERS
      SR 15,15 SET RETURN CODE
      BR 14 EXIT
*
* ERRORS AND MESSAGES
*
ERR1 MVC BUFFER(132),BLANKS
     MVC BUFFER(21),=C'Initial rewind failed'
     PUT PRINT,BUFFER
     B CLOSE
ERR2 MVC BUFFER(132),BLANKS
     MVC BUFFER(12),=C'Write failed'
     PUT PRINT,BUFFER
     B CLOSE
ERR3 MVC BUFFER(132),BLANKS
     MVC BUFFER(22),=C'Write tape mark failed'
     PUT PRINT,BUFFER
     B CLOSE
ERR4 MVC BUFFER(132),BLANKS
     MVC BUFFER(20),=C'Second rewind failed'
     PUT PRINT,BUFFER
     B CLOSE
ERR5 MVC BUFFER(132),BLANKS
     MVC BUFFER(17),=C'Read failed '
     PUT PRINT,BUFFER
     B CLOSE
ERR10 WTO 'NO TAPE DD STATEMENT'
      B RETURN
ERR11 WTO 'NO SYSPRINT DD STATEMENT'

```

```

        B      CLOSE
NOINPUT WTO   'NO PARAMETER OR SYSIN FOUND'
        B      RETURN
*
*  CONSTANTS, WORK AREAS
*
SAVEAREA DC   18F'0'
SEC2     DC   F'200'          TWO SECONDS
STAPEO   DCB   DSORG=PS,MACRF=(E),DDNAME=TAPE
PRINT    DCB   DSORG=PS,DDNAME=SYSPRINT,MACRF=(PM),LRECL=132,      X
          DCB   BLKSIZE=13200,RECFM=FB
SYSIN    DCB   DSORG=PS,DDNAME=SYSIN,MACRF=(GM),LRECL=80,        X
          DCB   RECFM=FB,EODAD=D
          DS    D'0'
SCCWRITE DC   X'01',AL3(BUFFER),X'20',AL3(100)
SCCWTM   DC   X'1F',AL3(0),X'20',AL3(1)
SCCREW   DC   X'07',AL3(BLANKS),X'20',AL3(1)
SCCREAD  DC   X'02',AL3(BUFFER2),X'20',AL3(3200)
SECB     DC   F'0'
SIOB     DC   X'42000000'
          DC   A(SECB)          A(ECB)
          DC   2F'0'           CSW
          DC   A(0)            A(CCW)
          DC   A(STAPEO)       A(DCB)
          DC   2F'0'
*
PARM     DC   CL100' '          PARM CAN BE UP 100 CHARACTERS
PARMFLG  DC   X'00'           00=PARM, FF=SYSIN
MOVPARM  MVC   BUFFER(0),2(1)  USED BY EX INSTRUCTION
BLANKS   DC   CL132' '
BUFFER   DC   CL132' '
          LTORG
BUFFER2  DS   CL4000' '
          END
/*
/**.SYSLMOD DD DISP=OLD,DSN=OGDEN.LIB.LOAD(AWSCMDX)
//G.TAPE DD UNIT=(580,,DEFER),LABEL=(1,BLP),VOL=SER=123456,DSN=X
//G.SYSPRINT DD SYSOUT=*
//G.SYSIN DD *
tee xfile2
This is a line
This is line 2
This is line 3
/*

```





## Problem handling

There are several aspects to 1090 problem handling, including the following:

- ▶ Problems starting 1090 operation
- ▶ Problems during 1090 operation
- ▶ Problems with emulated device files
- ▶ Problems with the underlying Linux system
- ▶ Problems with the System z operating system and applications

This chapter uses the term *zPDT service provider*. This may be an IBM Business Partner or some other zPDT provider that offers service. Not all zPDT users may have such service providers, and some of the information in this chapter may not apply in these cases.

The underlying Linux system, and whatever System z operating system and applications are being used, are not part of the 1090 and are not part of any 1090 support activity. 1090 support includes only the direct 1090 components. As a practical matter, there may be some overlap between Linux issues and 1090 problems, and you may need assistance from your zPDT service provider to isolate the problem.

## 8.1 Problems starting 1090 operation

These problems are most commonly related to devmap errors, and are often due to errors in Linux file names in the devmap. The solution is to check the devmap carefully, remembering that file names are case sensitive in Linux.

Problems obtaining a license (from the 1090 token) are typically due to an expired or unactivated token.

Messages about Time Cheat errors indicate a more serious problem. These are typically caused either by (1) moving a token between multiple PCs that do not have their time-of-day clocks closely synchronized, or (2) manipulation of the clock in the PC. See “Token dates and times” on page 2 for additional discussions about this.

The 1090 uses a number of Linux shared storage (virtual memory) areas. These are normally freed when the 1090 is ended with an **awsstop** command. A failure or incorrect handling during 1090 startup or operation might result in these shared storage areas not being released. This can prevent the 1090 from being started again. There are Linux commands to individually free shared storage areas, but this requires multiple detailed steps.<sup>1</sup> Rebooting Linux is a primitive but effective way to solve this particular situation.

## 8.2 Problems during 1090 operation

The 1090 maintains a number of logs and traces during operation. The 1090 programs may detect a problem and capture the logs/traces at the time of the problem. The user can also capture logs/traces with a **snapdump** command<sup>2</sup>. This command may be used when there is no indication from the 1090 that a problem exists, but the user detects a problem and may want to work with their zPDT service provider<sup>3</sup> to resolve the problem.

It is important to remember that this discussion is solely about 1090 operation. The **snapdump** data is not meaningful for addressing other problems, such as a problem with the System z operating system or System z applications.

A **snapdump** function typically creates a megabyte of data in `/home/ibmsys1/z1090/logs`, contained in various files. You may take as many **snapdumps** as you wish, remembering that each one takes space in the logs directory. Your zPDT service provider might want several, or one, or none of these dumps.

Files in the logs directory created by **snapdump** are retained until you remove them. Most other log and trace files in this directory are automatically deleted by the 1090 when appropriate. However, over time there may be a buildup of unwanted files in the logs directory. Assuming you are not working on an outstanding 1090 problem, you can simply delete all the files in the logs directory (doing this when the 1090 is not running). An easy way to do this is to use the **--clean** option of the **awsstart** command. Again assuming you are not working with a 1090 problem, you might use the **--clean** option every time you perform an **awsstart**. Conversely, you would not use the **--clean** option while you are working on a problem; some of the older log/trace files might be wanted at a later time.

The **senderrdata** command is used to package **snapdump** data into a tar file, which is typically a little less than a megabyte. This file can be sent to IBM, via your zPDT service

<sup>1</sup> The Linux **ipcrm** command can be used to remove shared resources that have been orphaned.

<sup>2</sup> The **snapdump** command is valid only while zPDT is operational.

<sup>3</sup> IBM internal users would communicate through the z1090 forum instead of through a zPDT service provider. Other users, without a defined service provider, might use another zPDT forum.

provider, or simply kept on your Linux system for potential use later. The `senderrdata` command can manage the FTP operation to IBM. These files (on the receiving system at IBM) are automatically deleted after a few days unless a formal problem (PMR or equivalent) event is opened; this can be done by your zPDT service provider.

Figure 8-1 on page 94 provides an overview of problem data handling. If `snapdump` data is sent to IBM (as outlined in the figure) then the `senderrdata` option to create a *configuration file* should also be used and the results sent to IBM.<sup>4</sup>

---

<sup>4</sup> Later 1090 versions may automatically combine the configuration data with the `snapdump` data.

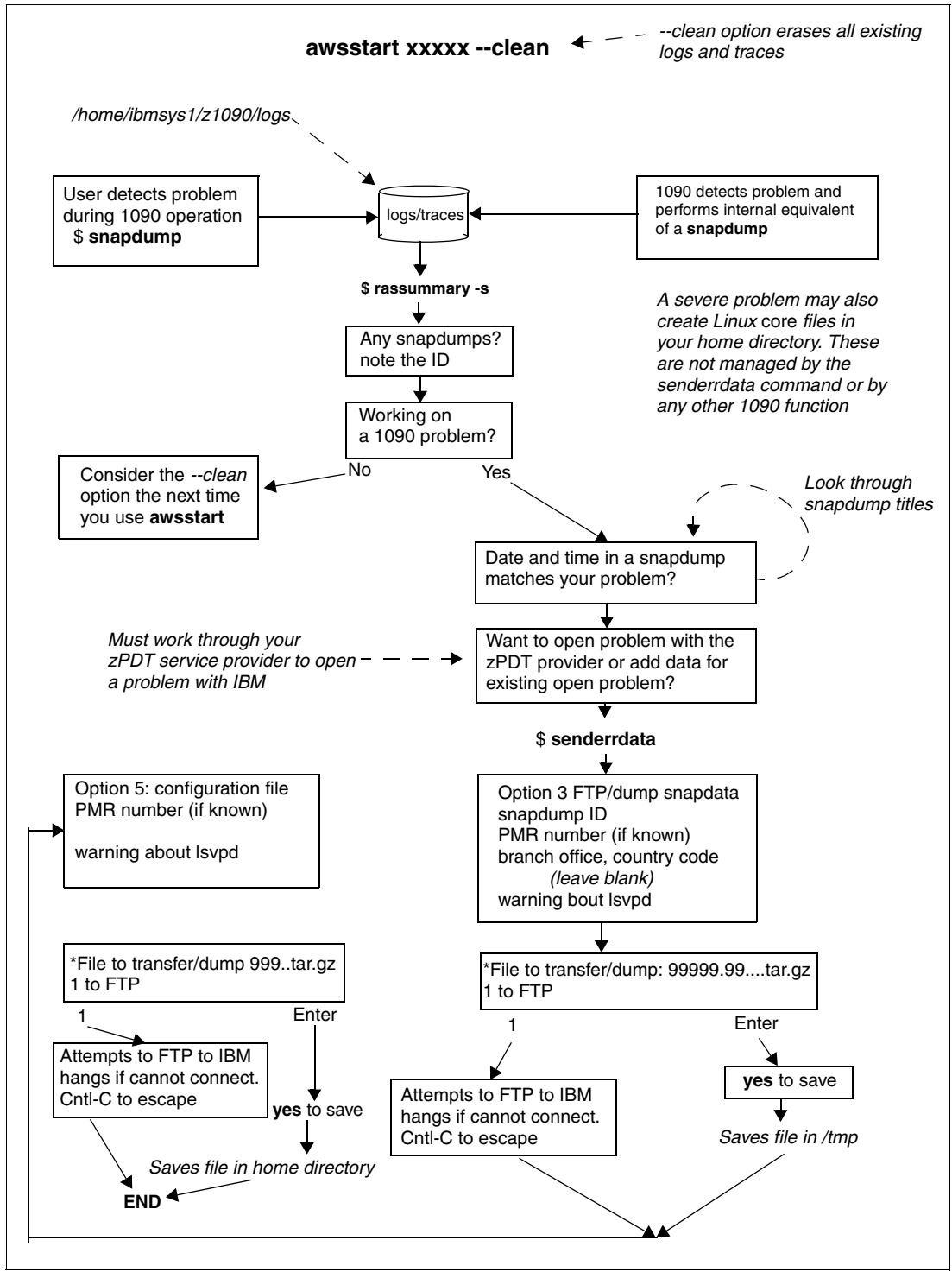


Figure 8-1 Problem data capture and reporting

If a problem incident is opened through your IBM Business Partner, you may be requested to send additional files. In general, once a problem incident is opened you should not delete anything from the logs directory.

If device managers fail they automatically create a trace file and are automatically restarted. This restart will not occur more than three times per minute. If a fourth failure occurs within

the same minute the device manager is not restarted and the devices it controls become *not operational* for the remainder of that 1090 session. The 1090 limits the size of the logs and traces and should never create more than about 30 MB per emulated device (and there is normally much less than this).

## 8.3 Core images

Severe problems may cause *core image* files to be created. If these are created by the 1090, they should go into the log subdirectory and be cleaned up with `--clean` option of `awsstart`. If you are actively working on a problem with IBM, these files may be useful. Otherwise they should be deleted because they can be rather large and might create a disk space problem.

Consistent dumps (“core images”) when the 1090 is started can occur if you have a relatively large number of emulated I/O devices (more than 100, for example) and you have not considered memory management adjustments. See “Devices, memory, msgmni, ulimit” on page 9 for more information.

The `snaddump` command is used when the 1090 is running. If you are unable to start the 1090 (with the `awsstart` command) or the 1090 ends immediately (before a `snaddump` can be taken), the problem may have created a core image file. In this case, the core image may help with problem analysis and should be preserved while the problem is under investigation.

## 8.4 Emulated volume problems

An emulated 3390 volume is a single Linux file that was created with the `alckd` command. Three variations of this command are useful for problem handling:

```
$ alckd /z/WORK03 -rs          scan emulated volume for format errors
$ alckd /z/WORK03 -rf        replace bad track with zeros
$ alckd /z/WORK03 -r         display volser and size
```

The `-rs` function scans the emulated volume and verifies that it is in the correct 1090 emulation format.<sup>5</sup> The `-rf` function replaces improperly formatted tracks with a properly formatted track containing zeros. The original contents of the track are lost, but the functionality of the volume is maintained.

Assuming that your emulated 3390 volumes are in the `/z` directory and there are no other file types in this directory, you could verify the format of all the volumes with the following Linux shell commands:

```
$ cd /z                      (location of emulated ckd, and nothing else)
$ for i in *; do alckd $i -rs; done
```

The `ckdPrint` command can be used to examine the contents of an emulated CKD volume. This command prompts for the beginning cylinder and head numbers and the ending cylinder and head numbers. These numbers are in decimal. The following example lists the records on cylinder 0, head 0 of volume Z9SYS1:

```
$ ckdPrint /z/Z9SYS1
DeviceType 3390, Cylinders-3339, Trks/Cyl-15, TrkSize-56832
Input extent in decimal - CC-low HH-low CC-high HH-high
00 00 00 00
```

<sup>5</sup> Only the emulation format is checked. There is no check for data content or operating system metadata (label, VTOC, and such).

....  
....

The **tapeCheck** command may be used to verify the format of an awstape file. That is, this command reads the awstape file and verifies that the awstape control blocks are logically correct.

### Special problem-related commands

The **senderrdata** command provides a menu of options:

1. rassummary (uses the Linux less command)
2. rassummary -s (for an overview of snapdump incidents)
3. FTP/dump snapdump data
4. FTP/dump PE directed files (used only at IBM request)
5. Create configuration information file
6. Logs Directory maintenance
7. FTP/dump rassummary created files
8. FTP/dump all files in logs directory
9. snapdump

The *dump* options in this menu mean that a tar file is created, containing the selected files, and the tar file is saved in /tmp. The dump option (to create a tar file) is especially useful if the 1090 machine is not connected to the Internet.

A number of 1090 commands are intended to be used only at the direction of IBM support personnel and they will supply the specific commands and operands to be used. This category includes the following commands:

- \$ awslog (including --logsize and --logcount operands)
- \$ doOSAcmd (various subcommands)
- \$ dreg (shared resource registry)
- \$ dshrmem (shared memory)
- \$ printlog (only for some .gz logs)
- \$ printtrace (only for some .gz traces)

The contents and formats of the various log and trace files are not documented and are intended only for diagnostic use. Our experience is that these files are not useful for solving general user-level problems.

## 8.5 Linux monitoring

Some monitoring of Linux statistics may be helpful. The **vmstat** command is useful and causes very little interference with other processes in Linux.

```
$ vmstat 3 4 (4 samples at intervals of 3 seconds)
procs -----memory----- --swap-- ----io----- --system-- ----cpu-----
r b swpd free buff cache si so bi bo in cs us sy id wa
2 0 0 397600 101936 2398160 0 0 109 52 154 350 14 2 81 3
```

- r = number of Linux processes waiting for run time
- b = number of Linux processes sleeping
- swpd = amount of swap space used (KB)
- free = amount of idle memory (KB)
- buff = amount of memory used as buffers (KB)
- cache = amount of memory used as cache (KB)
- si, so = Linux paging activity in KB/second

bi, bo = I/O activity in blocks/second  
 in = interrupts per second  
 cs = context switches per second  
 us = percent of CPU time in user mode  
 sy = percent of CPU time in kernel mode  
 id = percent of CPU time idle  
 wa = percent of CPU time waiting for I/O

Important data from **vmstat** includes the Linux swap rates (si, so); any number here can degrade 1090 performance. If the 1090 suffers a page fault, then the whole System z CP operation must wait for the page fault to be resolved. The wa statistic (CPU percentage waiting for I/O) provides an indirect indication of I/O overload.

The **top** command provides data about individual Linux processes. We are most interested in the emily process (which is the name of the Linux process representing the CP), but information about various device manager processes can be useful in spotting bottlenecks. Enter **q** from the keyboard to terminate the **top** command.

```

$ top
top - 11:14:14 up 1:24,  3 users,  load avg: 1.26, 1.15, 1.02
tasks: 128 total, 1 running, 121 sleeping, 6 stopped, 0 zombies
CPU(s): 49.9% us, 0.2% sy, 0.0% ni, 49.7% id, 0.0% wa 0.0% hi, 0.0% si
mem: 3111660k total, 2719204k used, 392456k free, 105356k buffers
swap: 2104504k total,      0k used, 2104504k free, 2397740k cached

  PID  USER   PR   NI  Virt  RES  SHR  S  %CPU  %MEM  TIME  COMMAND
 7040  ibmsys1 25    0 1549m 1.5g 1.5g  S   100  49.6 35:41.2  emily
 7051  ibmsys1 16    0 1510m 36m  36m  S    0   1.2  0:03.1  awsckd

```

The **XOsview** program provided with openSUSE Linux, if installed, can be started from **System** → **Monitor** → **XOsview**. It provides a very dynamic display of individual processor usage, memory usage, paging, and so forth. This program uses X11 graphics and creates more overhead than **vmstat** or **top**. If you are concerned about paging exposures on your system, we suggest that **XOsview** be started before starting the 1090.

The **XOsview** program, by default, uses a split line to display PC processor activity. Half of the line indicates instantaneous activity and the other half of the line shows an average rate with a decay factor averaged over several seconds. The decay presentation may be removed as follows:

```

# cd /usr/lib/X11/app-defaults
# gedit XOsview

```

Change `*cpuDecay` from True to False.





# Linux for System z

Linux for System z is not an IBM product and is not distributed by IBM. Several Linux products and distributions can be considered under the generic name of “Linux for System z.” This chapter describes two specific installation paths. Many other distributions and installation paths are possible, and we make no attempt to describe all ways and styles for installing Linux for System z.

In both our examples, most of the file names, sizes, and device addresses selected are arbitrary.

## 9.1 SUSE Linux Enterprise Server

We installed SUSE Linux Enterprise Server<sup>1</sup> 10 (with service pack 3) as the base operating system on zPDT. That is, we did not install it as a guest under z/VM. We first attempted to install SLES 11, but were unable to adapt it to the particular installation method we wanted to use.<sup>2</sup>

We created two 3390 volumes for SLES. One was a 3390-9 volume, intended to hold all the installed system with considerable space left over for user data. The other was a 3390-1 volume intended for swap space. Both these sizes (-9 and -1) are arbitrary. The installed SLES required about 2.8 GB, which is slightly larger than a 3390-3.<sup>3</sup> We placed both emulated volumes in our /z directory, which is used in all our examples throughout these books.

```
$ alcckd /z/SUSE01 -d3390-9
$ alcckd /z/SUSE02 -d3390-1
```

We created a devmap (named suse390), as follows:

```
[system]
memory 3600M
3270port 3270                #not needed for this installation
processors 2

[manager]
name aws3274 0002            #3270s are not needed for this basic
device 0700 3279 3274       #installation and this stanza could be
device 0701 3279 3274       #omitted

[manager]
name awsckd 0101
device F00 3390 3990 /z/SUSE01
device F01 3390 3990 /z/SUSE02

[manager]
name awsosa 0019 --path=A0 --pathtype=OSD --tunnel_intf=y
device 404 osa osa
device 405 osa osa
device 406 osa osa
```

We started with a DVD containing the SLES 10 SP3 distribution for System z (which is typically stated as the “S390 version”). Our zPDT underlying Linux was openSUSE 11.0, and had **vsftpd** installed and operational. We began the installation process by using the zPDT command that IPLs from specially formatted CD or DVD volumes. This action invokes a dialog with the SCLP console (also known as the HMC console or the hardware console). When using zPDT, the **oprmsg** command is required to send replies to this console; these illustrate the replies we used for our sample SLES installation:

```
$ ip1_dvd /media/SUSE-Linux-Enterprise-Server.001/suse.ins
Enter y to continue, n to end:g
y
```

<sup>1</sup> Commonly known as SLES.

<sup>2</sup> We wanted to install via an FTP server running on the underlying Linux with communication via a tunnel (tap0) connection. We were unable to interface the SLES 11 installation program to the tap0 interface.

<sup>3</sup> After being formatted for Linux, a 3390-3 has about 2.4 GB effective space. The 3390-9 has about 6.8 GB effective space.)

```

        (lots of output)
Main Menu
  1. Settings
  2. System Information
  3. Kernel Modules (Hardware Drivers)
  4. Start Installation or System
  5. Verify Installation CD-ROM / DVD
  6. Eject CD
  7. Exit or Reboot
  8. Power Off
oprmsg 4
  Start Installation or System
    1. Start Installation or Update
    2. Boot Installed System
    3. Start Rescue System
oprmsg 1
  Choose the Source Media
    1. CD-ROM
    2. Network
    3. Hard Disk
oprmsg 2
  Choose the Network Protocol
    1. FTP
    2. HTTP
    3. NFS
    4. SMB / CIFS (Windows Share)
    5. TFTP
oprmsg 1
  Please select the type of your network device
    1. OSA-2 or OSA Express
    2 Hipersockets
    -----
    3. Channel to Channel (CTC)
    4. ESCON
    5. Inter-User Communications Vehicle (IUCV)
oprmsg 1
  Please choose the CCW bus interface
    1. QDIO
    2. LCS
oprmsg 1
  Please choose the physical media
    1. Ethernet
    2. Token Ring
oprmsg 1
  Device address for read channel
oprmsg 0.0.0404
  Device address for write channel
oprmsg 0.0.0405
  Device address for data channel
oprmsg 0.0.0406
  Port name to use
oprmsg foobar
  Enable OSI Layer 2 support
    1. Yes
    2. No

```

```

oprmsg 2
    Enter the relative port number
oprmsg 0
    Automatic configuration via DHCP?
    1. Yes
    2. No
oprmsg 2
    Enter your IP address          (this is the future address of your SLES)
oprmsg 10.1.1.4
    Enter your netmask. For a normal class C .....
oprmsg 255.0.0.0
    If you need a gateway....If not, enter your IP address
oprmsg 10.1.1.4
    Enter the IP address of..name server...if none enter +++
oprmsg +++
    Enter the IP address of the FTP server
oprmsg 10.1.1.1          (the tap interface on the base Linux)
    Use a username and password?
    1. Yes
    2. No
oprmsg 1
    Enter the user name for the FTP server
oprmsg ibmsys1
    Enter the password for the FTP server
oprmsg xxxxxx
    Use a HTTP proxy?
    1. Yes
    2. No
oprmsg 2
    Enter the directory on the Server
oprmsg '/media/SUSE-Linux-Enterprise-Server.001'
    Select the display type
    1. X11
    2. VNC
    3. SSH
oprmsg 3
    (lots of output)
    starting hald (long pause)
    logon using 'ssh -X root@10.1.1.4'
    run 'yast' to start installation

```

At this point we opened another terminal window on our base Linux system and entered the specified commands:

```

$ ssh -X root@10.1.1.4
Are you sure you want to continue connection (yes/no)
yes
root@10.1.1.4's password: xxxxxx          (select a temporary password)
run yast to start the installation

```

At this point the terminal window is connected to the SLES installer program. Starting yast creates a semi-graphic display (“text mode”) that is used for the remainder of the installation. We do not attempt to duplicate these panel displays here, but we indicate the basic options we selected.

```

# yast          (this produces a very long pause while YaST2 starts)
English (Next) (select your language)

```

```

Yes, I agree (Next)      (Agree to the license statements)
Configure DASD disks
  Select both DASD units (one at a time)
  Perform Action: Activate
  Perform Action: Format
    (We suggest you do not format the two disks in parallel)
  (Next)
New Installation
Time Zone
Partitioning            (click on the title 'Partitioning')
  Create/edit to produce one partition on each DASD unit
    /dev/dasda()    6.8GB   IBM-DASD                0-10016
    /dev/dasda1    6.8GB   F Linux native ext3    /    0-10016
    /dev/dasdb()   782.5MB  IBM-DASD                0-1112
    /dev/dasdb1   782.5MB  F swap                  0-1112
  (Finish)
Software                (click on the title 'Software')
  (We took the defaults plus C/C++ and vsftpd)
  (Accept)

```

The installation started at this point and took 98 minutes in our case. At the end of this installation phase the SLES installer displays the following message:

```
Reboot and run /usr/lib/YaST2/startup/YaST2.ssh
```

and then reboots. This causes the ssh session to drop. In our case, the rebooted SLES was unable to start the OSA tap interface. We stopped zPDT (**awsstop**) and started it again (**awsstart**) and then IPLed SLES again (**ip1 F00**). The OSA interface started after doing this.

We used the second terminal window on our base Linux for the connection to SLES. First we needed to remove the existing ssh password file:

```

$ rm ~/.ssh/known_hosts4
$ ssh -X root@10.1.1.4
  (reply to message to create a new password for ssh connections to SLES)
  (You should now be connected to SLES)
# cd /usr/lib/YaST2/startup
# ./YaST2.ssh

```

This starts **yast** again (with a long pause during startup and a longer pause during “initializing catalogs”). The remaining yast phases included setting a root password, a host name, some network setup (we skipped the test option), setting the authentication method (local), and creating a userid (we created *ibmsys1*).

## 9.1.1 Comments

Once installed, SLES can be used by starting the 1090 (**awsstart**) and then **ip1 F00** (using the disk address in our example). Once IPLed, a connection via **ssh** is used to log into the SLES system.

```

(In a terminal window in your base Linux)
$ ssh root:10.1.1.4          (connect to SLES)

```

The addresses used in our example (F00, F01, 404, 405, and 406) are quite arbitrary and have no special meaning.

<sup>4</sup> This assumes that only the ssh password used for SLES installation was present.

SLES reported a number of OSA conditions during operation, but these did not appear to affect operation. The messages, appearing in the terminal window used to start zPDT, were similar to the following example:

```
TTL=64 ID=nnnnn DF PROT=TCP SPT=22 DPT=58132 Window=16024 RES=0x00 ACK
PSH URG P=0 OPT (0101080A000322xxxxxxxxxx)
```

```
SFW2-INext-Drop-DEF2T IN=eth0 OUT= MAC= SRC=10.1.1.4 DST=239.255.255.253 LEN=nn
```

We used a basic line mode terminal for initial work with SLES. An X terminal session may be started as follows:

```
(log into SLES root, using ssh as described above)
# vncpasswd (create a new password for vnc)
# vncviewer & (start the vnc server on SLES)

(open a new terminal window on the base Linux system)
$ vncviewer 10.1.1.4:5901 (connect to SLES)
```

## 9.2 Red Hat Enterprise Linux for System z under z/VM

This section describes one method of installing Red Hat Enterprise Linux for System z under z/VM on a zPDT system. Much of this material was described by Peter Holm of IBM Sweden. We stress that the following is merely one way of doing this, described in a specific environment. In particular, we worked with openSUSE 11.0 as our base Linux for zPDT; minor details may differ with other Linux distributions.

Before starting you need the following:

- ▶ A DVD containing Red Hat Enterprise Linux for System z. This is licensed material, and you should observe the relevant Red Hat license requirements. Our DVD was named RHEL\_5.3 s390x DVD.
- ▶ z/VM installed on the zPDT system, as described in the first part of this chapter.
- ▶ An operational FTP server in your base Linux. We had included **vsftp** in our base Linux installation, and we did the following:
  - Edit `/etc/vsftp.conf`

```
ftpd_banner="zPDT Base Linux FTP"
local_enable=YES
#listen=YES (disable the listen function)
```
  - Enable `inetd` to use `vsftp`, **Yast** → **Network Services (xinetd)** → **Enable** → **ftp (toggle status on)**.

We need to use ftp connections from a z/VM guest to the base Linux. We disabled the firewall (and the AppArmor® function) in our base Linux to permit these connections. Other base Linux distributions may need similar actions.

- ▶ A plan for IP addressing.

We used a z/VM VSWITCH function, and connected to the base zPDT Linux with a tunnel connection. We selected the following IP addresses:

- ▶ 10.1.1.1 for the base Linux end of the tunnel
- ▶ 10.1.1.2 reserved for other guests (not part of this exercise)
- ▶ 10.1.1.3 for z/VM TCP/IP
- ▶ 10.1.1.4 for Linux for System z (under z/VM)

These IP addresses are illustrated in Figure 9-1. Note that z/VM users of the VSWITCH all see an OSA device at addresses starting at 400. The OSA device defined for zPDT is at address 800, but this address is known only to the VSWITCH.

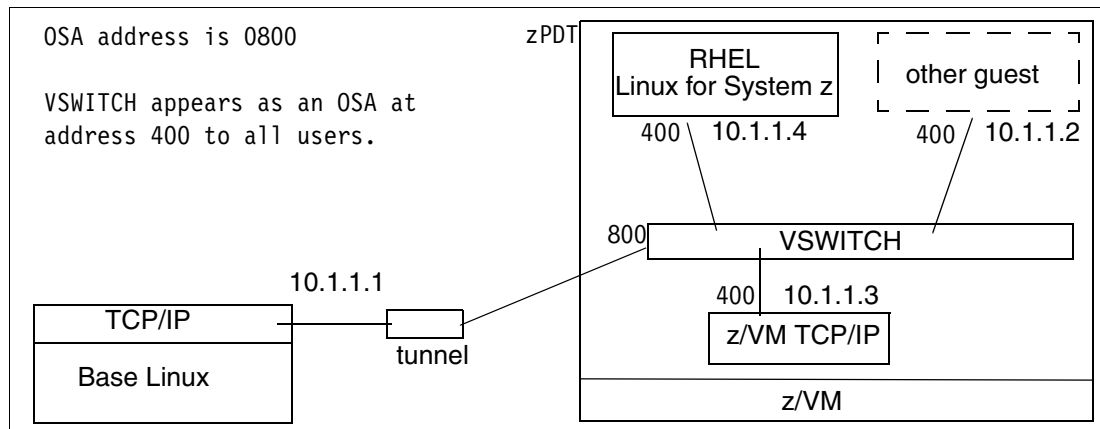


Figure 9-1 IP addressing

## Allocate volumes for Linux

We allocated four 3390-3 volumes for use by Linux for System z. The names and location of the emulated files are arbitrary. We do not need this many volumes; two would be sufficient.

```
$ alckd /z/ZLINUX1 -d3390-3
$ alckd /z/ZLINUX2 -d3390-3
$ alckd /z/ZLINUX3 -d3390-3
$ alckd /z/ZLINUX4 -d3390-3
```

## Create devmap

We used the following devmap. z/OS volumes are included for future use, but are not needed for the Linux work. The device addresses (device numbers) of the z/VM volumes, the Linux volumes, and the OSA tunnel device are arbitrary as is the number of 3270 devices, but the OSA addresses (starting at 800 in this example) are also specified in the z/VM VSWITCH setup.

```
[system]
memory 50000m          our base machine had 6 GB memory
3270port 3270
processors 2

[manager]
name aws3274 0002
device 0700 3279 3274 mstcon
device 0701 3279 3274
device 0702 3279 3274
device 0703 3279 3274      #You can add more 3270 devices if you want

[manager]
name awsckd 101
device 100 3390 3990 /z/540RES
device 101 3390 3990 /z/540W01
device 102 3390 3990 /z/540W02
device 103 3390 3990 /z/540PAG
device 104 3390 3990 /z/540SPL
#device 105 3390 3990 /z/PRODPK      #this volume not needed
```

```
[manager]
name awsckd 102
device 200 3390 3990 /z/LINUX1
device 201 3390 3990 /z/LINUX2
device 202 3390 3990 /z/LINUX3
device 203 3390 3990 /z/LINUX4
```

```
[manager]
name awsosa 0019 --path=A0 --pathtype=OSD --tunnel_intf=y
device 800 osa osa --unitadd=0
device 801 osa osa --unitadd=1
device 802 osa osa --unitadd=2
```

## Customize SYSTEM CONFIG

Log onto z/VM as user MAINT and perform the following functions:<sup>5</sup>

```
CPRELEASE A
LINK * CF1 CF1 MR
ACC CF1 Z
X SYSTEM CONFIG
  (make the following changes)
  System_Identifier_Default zPDT-VM      (maximum 8 characters; your choice)
  ...
  Operator_Consoles 0700 ,
                        System_3270 System_Console
  Emergency_Message_Consoles 0700 ,
                        System_Console
  DEFINE VSWITCH DTCVSW1 RDEV 0800
  MODIFY VSWITCH DTCVSW1 GRANT TCPIP
  MODIFY VSWITCH DTCVSW1 GRANT ZLINUX1
  MODIFY VSWITCH DTCVSW1 GRANT ZOS1
  ...
  CP_Access MAINT CF2 E RR      (add a drive letter to this line)
  CP_ACCESS MAINT CF3 F RR      (add a drive letter to this line)
ACC 193 X
CPSYNTAX SYSTEM CONFIG      (check syntax of your changes)
X LOCAL LOGO                (if you want to customize the VM logo)
RELEASE Z
LINK * CF1 CF1 RR
CPACCESS MAINT CF1 A SR
```

## Modify z/VM directory

First verify the location of minidisk space. We placed new minidisks on volume 540W02. Working as MAINT, do the following:

```
DISKMAP USER DIRECT C      (create minidisk map of z/VM volumes)
BROWSE USER DISKMAP A      (inspect the map)
```

Look for the last line for volume 540W02. It may be something like this:

```
LDAPSRV  191      3390      02366      02370      0005
```

<sup>5</sup> We assume a basic familiarity with XEDIT, and we do not include instructions for scrolling, saving files, and so forth. We show z/VM commands in upper case to distinguish them from other text; they may be entered in lower case on the terminal.

This indicates that a user (owned by LDAPSRV, which is not relevant) has a minidisk starting at cylinder 02366 and extending through cylinder 02370 (and is five cylinders long). This is the last minidisk on the volume. We can add minidisks starting with cylinder 02371. Your minidisk placement may be different; we assume cylinder 02371 in the following instructions.

```
X USER DIRECT C (edit the z/VM directory. Be careful!)
```

(Add the following entries; we placed them just before the MAINT entry.)

```
***** FOLLOWING ADDED BY OGDEN *****
*
USER ZLINUX1 ZLINUX1 1000M 1000M G
IPL CMS
MACH ESA 1
CPU 00 BASE
NICDEF 0400 TYPE QDIO DEVICES 3 LAN SYSTEM DTCVSW1
SPOOL 00C 2540 READER A
SPOOL 00D 2540 PUNCH A
SPOOL 00E 1403 A
CONSOLE 009 3215 T
LINK MAINT 190 190 RR
LINK MAINT 19D 19D RR
LINK MAINT 19E 19E RR
*OPTION LNKNOPAS APPLMON
MDISK 200 3390 DEVNO 200 MWV
USER DIRECT C1 F 80 43BLK 09/05/19 165/2183
===> BROWSE
MDISK 201 3390 DEVNO 201 MWV
MDISK 202 3390 DEVNO 202 MWV
MDISK 203 3390 DEVNO 203 MWV
MDISK 191 3390 2371 080 540W02 MR ALL (note the location of the mindisk)
*
***** END OF OGDEN ADDITIONS *****
```

Add a NICDEF line to the TCPIP entry in the directory:

```
*****
*
USER TCPIP TCPIP 128M 256M ABG
INCLUDE TCPCMSU
OPTION QUICKDSP SVMSTAT MAXCONN 1024 DIAG98 APPLMON
SHARE RELATIVE 3000
IUCV ALLOW
IUCV ANY PRIORITY
IUCV *CCS PRIORITY MSGLIMIT 255
IUCV *VSWITCH MSGLIMIT 65535
***** FOLLOWING LINE ADDED BY OGDEN*****
NICDEF 0400 TYPE QDIO DEVICES 3 LAN SYSTEM DTCVSW1
LINK 5VMTCP40 491 491 RR
...
```

Repeat the DISKMAP commands to verify that your new minidisks are correctly defined:

```
DISKMAP USER DIRECT C (create minidisk map of z/VM volumes)
BROWSE USER DISKMAP A (inspect the map)
```

Look for GAP or OVERLAP comments in the DISKMAP. These may indicate errors, although a GAP does no harm.

Lastly, activate the new directory:

```
DIRECTXA USER DIRECT C
```

### Customize TCPIP

Logon as user TCPMAINT and customize the TCPIP profile. Use the command X PROFILE TCPIP D to access the profile on the D minidisk of user TCPMAINT. The following listing has removed all the comments in the original profile; this is not necessary and was done to reduce the size of the listing. The important updates are to the DEVICE, LINK, HOME, and GATEWAY statements.

```
...
...
DEVICE OSAQDI01 OSD 0400 PORTNAME OSA1 PRIROUTER AUTORESTART
LINK ETH0 QDIOETHERNET OSAQDI01

ARPAGE 5

HOME
10.1.1.3 255.0.0.0 ETH0

GATEWAY
DEFAULTNET 10.1.1.1 ETH0 1492

TRANSLATE

START OSAQDI01
```

### Add AUTOLOG IDs

Logon as AUTOLOG1 and edit PROFILE EXEC A as follows:

```
/* AUTOLOG1 PROFILE EXEC */
Address Command
'CP XAUTOLOG VMSERVS'
'CP XAUTOLOG VMSERVU'
'CP XAUTOLOG VMSERVR'
'CP XAUTOLOG DTCVSW1'
'CP XAUTOLOG TCPIP'
'CP ENABLE ALL'
'CP RECORDING ALL OFF'
'CP EXEC TELL OP **RECORDING ALL OFF** WAS EXECUTED'
```

### Operational check

At this point we shut down and reIPLed z/VM. This should exercise the AUTOLOG machines and generally provide a clean start. To access TCP/IP functions from various userids you need to do the following link and acc commands. We logged on as MAINT and tried the following.

```
LINK TCPIP 592 592 RR
ACC 592 F (or some other drive letter if F is busy)

NETSTAT DEV (try a NETSTAT command)
PING 10.1.1.1 (PING the base Linux system)
```

If z/VM is running (with the customization described above) you should be able to ping it from your base Linux:

```
# ping 10.1.1.3
```

## Copy distribution DVD

This step is not required, but we found it to be convenient. We copied the complete Red Hat Enterprise Linux for System z to a temporary disk directory on the base Linux. This disk directory then becomes the target for subsequent ftp operations. On the base Linux system, mount the DVD and do the following:

```
$ mkdir /tmp/RHEL          (this directory name is arbitrary)
$ cd /media                (cd to the DVD)
$ cd R<tab>                (use tab function to complete name)
$ cp -R * /tmp/RHEL       (copy DVD. About 5 minutes)
```

## Obtain images

The next step is to obtain the Linux installation images from the DVD (or the copy of the DVD we placed in /tmp/RHEL). Log on to the ZLINUX1 userid and do the following:

```
FORMAT 191 B              (format the 80 cylinder minidisk)
    (make appropriate answer to format and label it; use any label)
LINK TCP/IP 592 592 RR    (enable use of TCP/IP by ZLINUX1)
ACC 592 E
FTP 10.1.1.1              (connect to base Linux)
    (logon as ibmsys1 or another convenient userid)
CD /tmp/RHEL/images      (Use disk copy of DVD)
LOCSITE FIX 80
BIN
GET kernel.img (repl
GET initrd.img (repl
ASCII
GET generic.prm (repl
QUIT
```

## Edit and create CMS files

Remain on the ZLINUX1 userid in z/VM. (The lines in these files all begin in the first column. The lines appear indented in the text below; this is done to separate the file contents from the z/VM commands. The italic comments in the text are not entered.) Edit the GENERIC PRM file as follows:

```
X GENERIC PRM
SET CASE MIXED           (this is an Xedit command)
                        (change the file to appear as shown here)
    root=/dev/ram0 ro ip=off ramdisk_size=40000
    CMSDASD=191 CMSCONFIG=redhat.conf
    vnc
```

The text in these files is sensitive to upper and lower case. If you have trouble entering lower case, use the SET CASE MIXED command for XEDIT.

Create a new file named REDHAT CONF as follows:

```
X REDHAT CONF
SET CASE MIXED           (this is an Xedit command)
                        (enter the following lines, starting in column 1)
    HOSTNAME="zlinux1.itso.ibm.com" (you may alter this line)
    DASD="200-203"
    NETTYPE="qeth"
    IPADDR="10.1.1.4"      (IP address for RHEL)
```

```

SUBCHANNELS="0.0.0400,0.0.0401,0.0.0402"      (VSWITCH addresses)
PORTNAME="FOOBAR"                             (information not used)
NETWORK="10.0.0.0"
NETMASK="255.0.0.0"
BROADCAST="10.1.1.255"
SEARCHDNS="ibm.com"                           (not used; you may alter)
GATEWAY="10.1.1.1"                             (IP address of base Linux)
DNS="10.1.1.1"
MTU="4096"

```

Double-check the file you entered; it is easy to make mistakes. Are all operands enclosed in quotes? Are the appropriate operands in lower case? Are the keywords spelled correctly? Next, create the file REDHAT EXEC as follows:

```

X REDHAT EXEC
SET CASE MIXED                               (this is an Xedit command)
                                           (enter the following lines, starting in column 1)

/* REXX */
'cl rdr'
'purge rdr all'
'spool pun * rdr'
'PUNCH KERNEL IMG A (NOH'
'PUNCH GENERIC PRM A (NOH'
'PUNCH INITRD IMG A (NOH'
'ch rdr all keep nohold'
'ipl 00c'

```

## Start the installation

The next step is to execute the REDHAT EXEC. You should still be logged onto ZLINUX1.

```

REDHAT                                         (this will execute the script)

```

This should produce a pause of several seconds and then three messages about RDR activity. This is followed by another pause (and much disk activity on the base machine). A number of startup messages appear; clear the z/VM panel as needed.<sup>6</sup> The following prompts must be answered. If you receive additional prompts, you probably have an error in the REDHAT CONF file. You can correct it and rerun the REDHAT EXEC script.

```

Enter the mode of operation for the OSA device
0 for layer 3 mode (default)
1 for layer 2 mode
0
Enter the port number for the OSA device
0 for port number 0 (default)
1 for port number 1
0
The authentication of host.....
Continue? (yes/no)
yes

```

This should be followed by a message indicating you can telnet to ssh to the Red Hat installer. Open a window on the base Linux system:

```

$ su                                           (switch to root)
# ssh 10.1.1.4                                (connect to Red Hat installer)

```

<sup>6</sup> PA2 can be used to clear the panel, or the Pause key can be used if the x3270 keymap has been modified as we suggested during zPDT installation.

An installer page (from the Anaconda program) should appear in a new window on the base Linux. You must respond to several initial questions. Use the tab key, up/down arrow keys, and the space bar to work with these panels:

```
Choose a Language: English <OK>
Installation Method: FTP <OK>
  FTP Site: 10.1.1.1
  RHEL Directory: /tmp/RHEL/
  [*] use non-anonymous ftp <OK>
    Further FTP setup:
      Account name: ibmsys1           (for example; a userid on the base Linux)
      Password: xxxxx                (supply appropriate password)
The VNC server is now running.
Please connect to 10.1.1.4:1 to begin installation.
Press <enter> for a shell
Starting graphical installation
<Enter>
```

## Main installation

In a terminal window on the base Linux enter:

```
$ vncviewer 10.1.1.4:1
```

This should produce a window with basic graphics. Work through the installation questions; your choices may differ from ours.

```
Installation number: Skip
Initialize 3390 volume(s): yes           (about 8 minutes each)
Partitioning:                               (we used the defaults)
Network devices:                             (we used the defaults)
  (No primary DNS in our case)
Time:                                         (as appropriate)
Root password: xxxxx
Customize software:                          (we used the defaults)
  (Long pause)
Begin installation:
...
Reboot
```

This final phase of the installation took about 2 hours 30 minutes. Eventually, RHEL will offer to reboot. You can select this. Later you can boot RHEL by simply entering an **ip1 200** command while in the ZLINUX1 userid of z/VM.

## Initial RHEL operation

RHEL offers to boot in two modes; we are unclear about the usefulness of this option:

```
zIPL v1.5.3 Interactive boot menu
  0 default (linux)
  1 linux
```

This message times out after 15 seconds and takes option 0. Booting RHEL (from the ZLINUX1 userid) creates a logon panel on the VM terminal. This is in 3215 mode and is not a very useful terminal, especially for using **vi**. The following steps provide a better terminal on the base Linux desktop.

```
(log in to RHEL as root, using the 3215 session)
# /etc/init.d/iptables save           (only needed once)
# /etc/init.d/iptables stop          (turn off the firewall)
```

```
# vncpasswd (create a vnc password)
# vncserver & (start the vnc server)
```

In a window on the base Linux desktop:

```
$ vncviewer 10.1.1.4:5901 (5901 is default port for this vnc)
    (Enter the vnc password when requested)
```

This should open an Xwindows panel (in basic graphics mode) on the base Linux system. Use the left mouse button to obtain a menu for the panel contents.

## Graphics terminal

We found the vnc full-graphics mode to be rather slow. It can be created by using the basic terminal (on the vncviewer) to edit a file. Do not do this with the 3215-mode terminal (on the VM panel) because `vi` functions do not work correctly with this terminal.

```
# cd /root (home directory for root)
# cd .vnc
# vi xstartup (edit this file)
    (uncomment the two indicated lines and then save the file)
    unset ....
    exec ....
```

You need to exit from the vnc terminal (use `Ctrl-d` to close windows), close the whole vnc window on the base Linux. Then stop the vncserver (using the 3215 panel) with:

```
# vncserver -kill:1
```

If this command does not work for some reason, you may want to shut down and reboot RHEL. Now start the vncserver again:

```
# vncserver &
```

In a base Linux window, start the vncviewer again:

```
$ vncviewer 10.1.1.4:5901 (5901 is default port for this vnc)
    (Enter the vnc password when requested)
```

The initial startup may be *very* slow; you may think the system has failed; be patient. Eventually you should see a graphic window for RHEL. The “normal” icons should be present. We suggest you defeat SELinux using the path **System** → **Administration** → **SELinux Management** and set the mode to *permissive*. Initial use of the icons and paths is slow, but it becomes faster once they are used.

## Stopping RHEL

Use a `shutdown -h now` command (from the 3215 terminal or from a vnc session) to stop RHEL.



## LAN notes

A basic LAN setup, for TCP/IP using QDIO interfaces, is covered in the second volume in this document series. This chapter discusses additional options and usage notes.

## 10.1 Recent changes

Three changes that affect LAN/OSA usage are available in zPDT release 39.16 (and later releases). These are:

- ▶ Alias Ethernet devices in the base Linux (such as eth0:0) may be defined and used by the awsosa device manager.
- ▶ Multiple tunnel OSA device managers may be used. These would have associated CHPID numbers of A0, A1, A2, and so forth and would be identified as tap0, tap1, tap2, and so forth.
- ▶ Ethernet adapters with Linux names such as wlanx may be used (however, we retain the general caution about using wireless interfaces).

Starting with the zPDT release available in June 2010, the following four tunnel interfaces are available:

CHPID	LinuxName	default IP address	default IP mask
A0	tap0	10.1.1.1	255.255.255.0
A1	tap1	10.1.2.1	255.255.255.0
A2	tap2	10.1.3.1	255.255.255.0
A3	tap3	10.1.4.1	255.255.255.0

If you are using a single tunnel interface, as provided in previous zPDT releases, no changes are required. However, we suggest that you change the IP mask in your z/OS PROFILE to 255.255.255.0 to allow for potential use of multiple interfaces. Starting with this release, the **find\_io** command may show all four interfaces, even if you have defined only one in your devmap.

In general, the multiple tunnel interfaces are intended for use with multiple zPDT instances.

The default IP address and mask may be changed by parameters in the name statement associated with the OSA definition in your devmap.

At the time of writing, Linux bonding of several LAN adapters into a single virtual adapter is not usable with zPDT. (This could change in the future; be certain to check the most recent zPDT documentation.)

## 10.2 Non-QDIO operation

When using the non-QDIO interface to the emulated OSA-Express2 function, the key parameters might look like the following:

Devmap

```
[manager]
name awsosa 22 --path=F0 --pathtype=OSE
device E20 osa osa --unitadd=0
device E21 osa osa --unitadd=1
```

z/OS TCP/IP Profile

```
DEVICE LCS1 LCS E20 AUTORESTART
LINK ETH1 ETHERNET 0 LCS1
HOME 192.168.0.61 ETH1
```

```

...
BEGINRoutes
; Destination Subnet Mask FirstHop Link Size
ROUTE 192.168.0.0 255.255.255.0 = ETH1 MTU 1492
ROUTE DEFAULT 192.168.0.1 ETH1 MTU DEFAULTSIZE
ENDRoutes
...
START LCS1

```

This example assumes that z/OS contains an appropriate CTC or OSA definition for addresses E20 and E21.<sup>1</sup> Different addresses could be used, of course, but they must match the IODF in your z/OS system. The HOME address and ROUTE statements in the example are just examples, of course. The GATEWAY statements could be used instead of the ROUTE statements. The `--unitadd` parameter is used in the devmap because the default OSA unit addresses<sup>2</sup> would be 20 and 21 (using the *two* low-order digits of the device number) and we want unit addresses 0 and 1.<sup>3</sup>

## 10.2.1 More complete example

We used the `find_io` command to determine that our Ethernet adapter was eth0, and that it was assigned as CHPID F0. (As a practical matter, the integrated Ethernet adapter is always assigned to CHPID F0.) The tunnel interface is always CHPID A0. We elected to use the QDIO mode for both OSA interfaces, although the listings also show non-QDIO settings as comments. We used the following devmap:

```

[system]
memory 1600m
3270port 3270
processors 2

[manager]
name aws3274 0002
device 0700 3279 3274 mstcon
device 0701 3279 3274 tso
device 0702 3279 3274 tso
device 0703 3279 3274 tso

[manager]
name awsckd 0001
device 0A80 3390 3990 /z/ZBRES1
device 0A81 3390 3990 /z/ZBRES2
device 0A82 3390 3990 /z/ZBSYS1
device 0A83 3390 3990 /z/ZBUSS1
device 0A84 3390 3990 /z/ZBPRD1
device 0A85 3390 3990 /z/ZBPRD2
device 0A86 3390 3990 /z/ZBPRD3
device 0A95 3390 3990 /z/WORK01 #local volumes, not part of AD
device 0A96 3390 3990 /z/WORK02

# The following two stanzas for LCS operation

```

<sup>1</sup> LAN operation in LCS mode can use CTC definitions in the z/OS IODF. This is a carryover from earlier LAN implementations.

<sup>2</sup> This unit address is the (emulated) hardware address within the (emulated) OSA control unit. It is not the device number ("address" in common terminology).

<sup>3</sup> The default OAT used by OSA requires unit addresses 0 and 1 for TCP/IP when in OSE mode.

```

#[manager]
#name awsosa 0003 --path=F0 --pathtype=OSE
#device E20 osa osa --unitadd=0
#device E21 osa osa --unitadd=1

#[manager]
#name awsosa 0013 --path=A0 --pathtype=OSE --tunnel_intf=y
#device E22 osa osa --unitadd=0
#device E23 osa osa --unitadd=1

[manager]
name awstape 004
device 581 3490 3490

[manager]
name awscmd 1000
device 580 3490 3490

```

We used the following TCP/IP profile in z/OS. (If using the z/OS AD system, be certain to update the correct PROFILE. In the z/OS 1.11 AD system this is member PROF1.)

```

ARPAGE 5

DATASETPREFIX TCPIP

AUTOLOG 5
    FTPD JOBNAME FTPD1    ; FTP Server
    PORTMAP                ; Portmap Server
ENDAUTOLOG

PORT
    7 UDP MISCSERV        ; Miscellaneous Server
    7 TCP MISCSERV
    9 UDP MISCSERV
    9 TCP MISCSERV
    19 UDP MISCSERV
    19 TCP MISCSERV
    20 TCP OMVS           NOAUTOLOG ; FTP Server
    21 TCP OMVS           ; FTP Server
    23 TCP INTCLIEN       ; Telnet Server
    25 TCP SMTP           ; SMTP Server
    53 TCP NAMESRV        ; Domain Name Server
    53 UDP NAMESRV        ; Domain Name Server
    69 UDP OMVS           ; OE TFTP SERVER
    80 TCP OMVS           ; OE WEB SERVER
    111 TCP PORTMAP       ; Portmap Server
    111 UDP PORTMAP       ; Portmap Server
    135 UDP LLBD          ; NCS Location Broker
    161 UDP OSNMPD        ; SNMP Agent
    162 UDP SNMPQE        ; SNMP Query Engine
    433 TCP OMVS          ; OE WEB Server
    443 TCP OMVS          ; Secure Server
    512 TCP RXSERVE       ; Remote Execution Server
    513 UDP OMVS          ; OE RLOGIN SERVER
    514 UDP OMVS          ; OE syslog server
    514 TCP RXSERVE       ; Remote Execution Server

```

```

515 TCP LPSERVE           ; LPD Server
520 UDP OROUTED          ; Routed Server
580 UDP NCPROUT          ; NCPROUTE Server
750 TCP MVSKERB          ; Kerberos
750 UDP MVSKERB          ; Kerberos
751 TCP ADM@SRV          ; Kerberos Admin Server
751 UDP ADM@SRV          ; Kerberos Admin Server
; 1021 TCP OMVS           ; OE FTP SERVER
1023 TCP OMVS            ; OE TELNET SERVER
1023 UDP OMVS            ; OE TELNET SERVER
1024 TCP OMVS            ; OE SERVICES
1415 TCP CSQ1CHIN        ; CSQ1 MQ TCP Listener
3000 TCP CICSTCP         ; CICS Socket
8879 TCP BBODMGR         ; SOAP JMX Connector port
7277 TCP BBODMGR         ; Cell Discovery port
9809 TCP BBODMGR         ; ORB port
9090 TCP BBODMGR         ; HTTP port
9043 TCP BBODMGR         ; HTTPS port
5755 TCP BBODMNC        ; Daemon port
5756 TCP BBODMNC        ; Daemon SSL port

; WAS Base Node entries
8880 TCP BBOS001         ; SOAP JMX Connector port
;2809 TCP BBOS001        ; ORB port (COMMENT THIS ONE OUT)
9080 TCP BBOS001         ; HTTP port
9443 TCP BBOS001         ; HTTPS port
5655 TCP BBODMNB        ; Daemon port
5656 TCP BBODMNB        ; Daemon SSL port
; WAS ND Node entries
8879 TCP BBODMGR         ; SOAP JMX Connector port
7277 TCP BBODMGR         ; Cell Discovery port
9809 TCP BBODMGR         ; ORB port
9090 TCP BBODMGR         ; HTTP port
9043 TCP BBODMGR         ; HTTPS port
5755 TCP BBODMNC        ; Daemon port
5756 TCP BBODMNC        ; Daemon SSL port
; WAS Federation entries
8878 TCP BBON001         ; SOAP JMX Connector port
7272 TCP BBON001         ; Node Discovery port
2809 TCP BBON001         ; Node Agent's ORB port
9810 TCP BBOS001         ; Base Server's ORB port

SACONFIG DISABLED      ; This statement not in earlier releases

DEVICE LCS1  LCS      E20
LINK ETH1  ETHERNET  0  LCS1
HOME 192.168.0.61 ETH1

DEVICE LCS2  LCS      E22
LINK ETH2  ETHERNET  0  LCS2
HOME 10.1.1.2 ETH2

BEGINRoutes
;   Destination Subnet Mask   First Hop  Link  Size
ROUTE 192.168.0.0 255.255.255.0   =         ETH1 MTU 1492

```

```

ROUTE 10.0.0.0    255.0.0.0          =      ETH2 MTU 1492
ROUTE DEFAULT    192.168.0.1        ETH1 MTU DEFAULTSIZE
ENDRoutes

ITRACE OFF

IPCONFIG NODATAGRAMFWD

TCPCONFIG RESTRICTLOWPORTS

UDPCONFIG RESTRICTLOWPORTS

START LCS1
START LCS2

```

### 10.3 Local routers and DHCP

In most cases the 1090 user has a single network Ethernet cable interface available, probably connected to a router somewhere external to the user. This external network interface typically expects a DHCP client, and this presents two problems:

- ▶ The System z operating system (z/OS, for example) might not operate as a DHCP client. That is, it may want a fixed IP address. In general, network-connected users do not have fixed IP addresses.
- ▶ The 1090 machine may have multiple LAN adapters, requiring multiple network connections.

The second book in this series describes a number of ways to connect z/OS (or other System z operating systems) to larger networks. Another option is to use a small router, as shown in Figure 10-1.

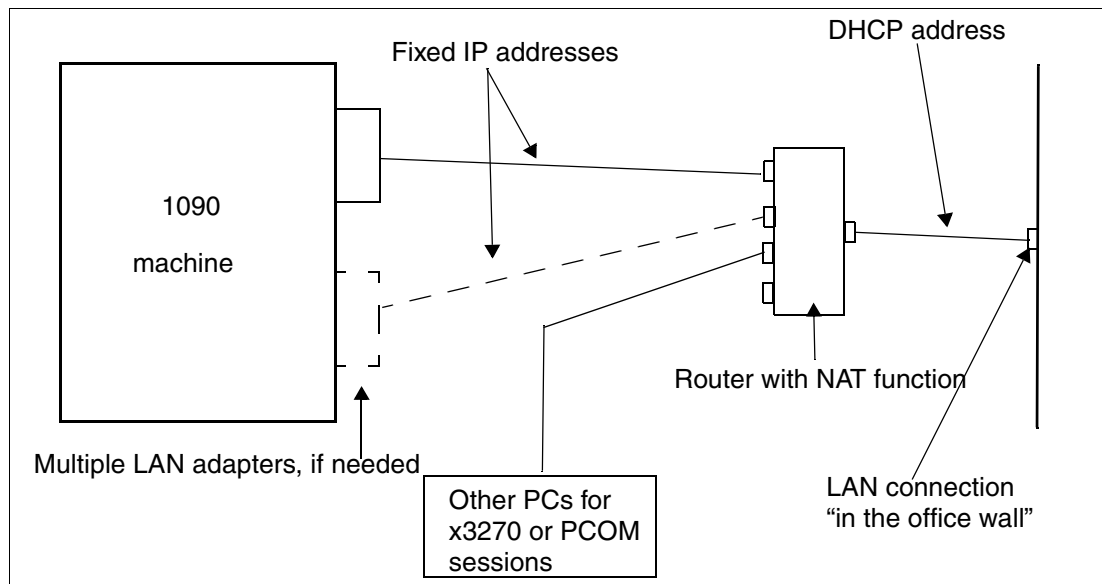


Figure 10-1 Small hub or router usage

The Network Address Translation function in the router allows your machine to work with fixed IP addresses (provided by the NAT router).<sup>4</sup> These are typically in the 192.168.xxx.xxx range.

The router, in turn, works with variable DHCP addresses provided by your external network. If NAT is not required, the router in the illustration can be changed to a simple hub. (The hub option assumes the LAN connection can operate with a hub; this is not always the case and you may need help from your network administrators to determine the best configuration.)

We specified the base router IP address (192.168.0.1) as the default gateway address in our TCP/IP definitions (for both Linux and z/OS). For a multiuser system we connected additional PCs to the router (which supplied its own range of DHCP addresses, if requested). The additional PCs can connect to aws3270 (using the Linux IP address and port 3270) or to OSA (using the IP address assigned, specified in the z/OS TCP/IP PROFILE).

Most routers can be configured to pass incoming port connections to specific local IP addresses. This requires some work with the router software, but allows the handling of incoming connections to z/OS (coming from a DHCP-based external network).

## 10.4 Shared Ethernet adapters

Scenario 4, in the LAN descriptions in the second volume of these books, uses a single “real” Ethernet adapter (eth0, in the base Linux) for both Linux and z/OS. Some users find this confusing. Figure 10-2 illustrates this usage in more detail.

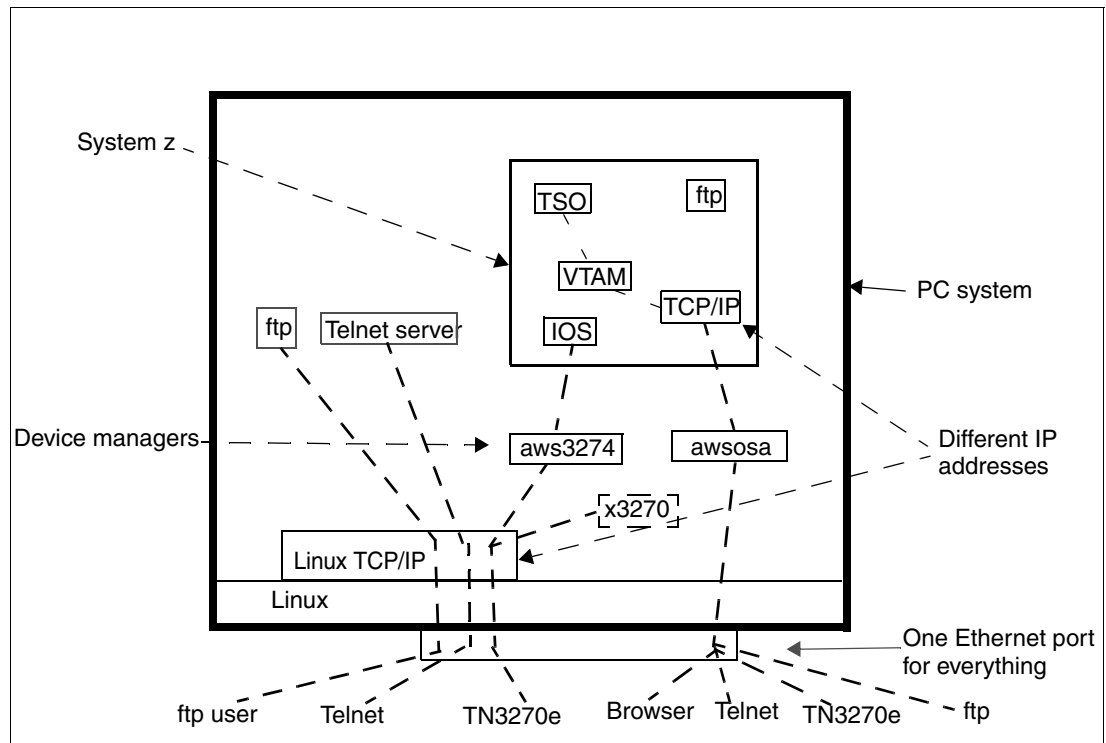


Figure 10-2 Shared Ethernet adapter

The awsosa device manager is independent from Linux TCP/IP, although it can use the same Ethernet adapter. Up to 16 TCP/IP stacks (in z/OS or z/VM) can connect to the awsosa device manager. Each of these TCP/IP stacks defines its own IP address. The configuration shown in Figure 10-2 would have two IP addresses, one for Linux TCP/IP and one for z/OS TCP/IP. These IP addresses are unrelated. External routing rules typically require that both IP

<sup>4</sup> The router might also function as a DHCP server, providing DHCP addresses in a portion of its address range.

addresses be on the same subnet, but this rule is external to the 1090. (Many examples in this documentation series use 192.168.0.60 for the Linux IP address and 192.168.0.61 for the z/OS IP address.)

The system in this illustration provides two paths for a user to connect to z/OS TSO. One path is through Linux TCP/IP and the aws3274 device manager. The other path is through the awsosa device manager and z/OS TCP/IP. (In this illustration, the awsosa port could be operating in either QDIO or non-QDIO mode.)

Note that there is no connection between OSA and the base Linux in this situation. It is a Linux design oddity that the two users of the physical Ethernet interface cannot communicate with each other.

Note these differences:

- ▶ The first path mentioned does not involve z/OS TCP/IP. The TN3270e client connects to the IP address used by the base Linux system. The client also specifies the port number assigned to the aws3274 device manager; this is port 3270 in our examples. z/OS accepts the connection as a coax-attached local 3270 terminal and is unaware that the client is actually connected via Linux TCP/IP. For this operation the z/OS OSA and TCP/IP functions need not be implemented. The MVS console must use this path.
- ▶ The second path uses a different IP address for OSA connections (assigned by z/OS PROFILE statements, or equivalent). z/OS TCP/IP internally passes TN3270e client connections to VTAM and thence to TSO. z/OS TCP/IP can also manage ftp sessions, telnet sessions, and so forth. The MVS console cannot use this path.

Figure 10-3 extends this concept to include a tunnel connection between z/OS and the base Linux.

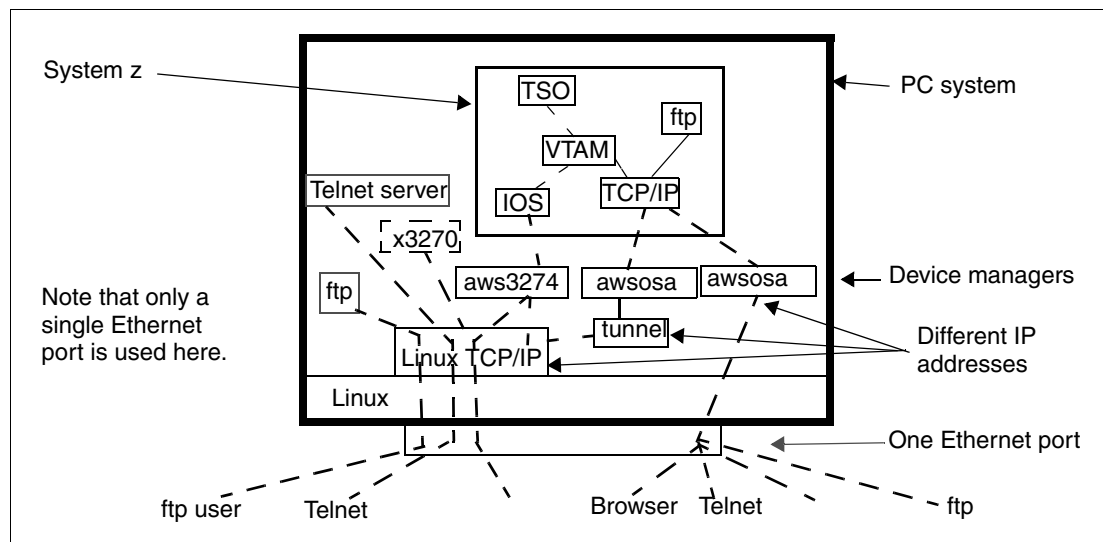


Figure 10-3 Shared Ethernet and tunnel

The tunnel environment allows connections between Linux TCP/IP applications (such as ftp, telnet, and x3270) and OSA TCP/IP applications<sup>5</sup> (such as ftp, the TN3270e server that is part of z/OS communications manager, and so forth). The tunnel environment creates a

<sup>5</sup> A more exact statement would reference TCP/IP applications within an operating system that is using the OSA-Express2 interface, of course. In our examples this would be z/OS applications (such as the TN3270e server) using the z/OS TCP/IP stack that interfaces to OSA-Express2. We abbreviate this detail by simply referring to an OSA application.

virtual adapter similar to an Ethernet adapter. This virtual adapter is assigned its own IP address on both the Linux and OSA side, as illustrated in Figure 10-3.

We strongly recommend that the tunnel IP addresses (for the Linux side and the OSA side) be on a subnet separate from any other IP addresses involved in the system. We emphasize this in our documentation by using 10.x.x.x addresses for the tunnel and 192.168.x.x addresses for other connections.

## 10.5 Base Linux LAN notes

Messages such as the following:

```
SFW2-INext-DROP-DEFAULT IN=tap0 OUT= MAC= SRC=10.1.1.1 .....
```

may be seen in the Linux log (with a `dmesg` command). These messages are related to the use of multicasting when looking for a DNS name server. The source address indicated in the message (10.1.1.1) is associated with a tunnel (tap) device in typical zPDT operation and is unlikely to find a DNS server.

These messages do no harm. If your Linux system has no need to find a DNS server (on any LAN interface) you can eliminate the messages by editing `/etc/host.conf` and changing *multi on* to *multi off*.





## DASD volume migration

The 1090 package includes a client-server utility for moving 3380/3390 volumes from a remote z/OS or z/VM system to zPDT. The server portion of this utility runs under z/OS or z/VM on the remote System z. This is typically a large System z, but it could be another zPDT system. The client portion runs under the base Linux on your zPDT machine.<sup>1</sup> The client and server are connected via TCP/IP.

This utility is especially useful when transferring many volumes to a zPDT system.

The server portion (on the remote z/OS or z/VM) reads all the tracks on a selected volume and sends them to the client (on the local base Linux). The client transforms it into the emulated 3380/3390 format used by zPDT and writes it as a Linux file. You then use this file as an emulated volume under zPDT.

One volume is processed for each client command that is sent to the server; you can create a Linux script with multiple invocations. The server portion (on z/OS) requires specific RACF authorizations. It can copy active volumes, although the usefulness of the copy might be questionable, depending on the volume activity at the time. The z/VM version requires that the server users have access to the full volumes being sent.

The speed of the copies depends on the TCP/IP bandwidth between the client and server, and the contents of each track. A considerable amount of data is involved on a typical 3390 volume; the transmission may take some time. A restart function is provided to continue a failed transfer.

A conceptual overview is shown in Figure 11-1.

<sup>1</sup> zPDT need not be operational while this utility is being used. Due to expected LAN and disk activity, it is probably better to use the migration utility when zPDT is not active.

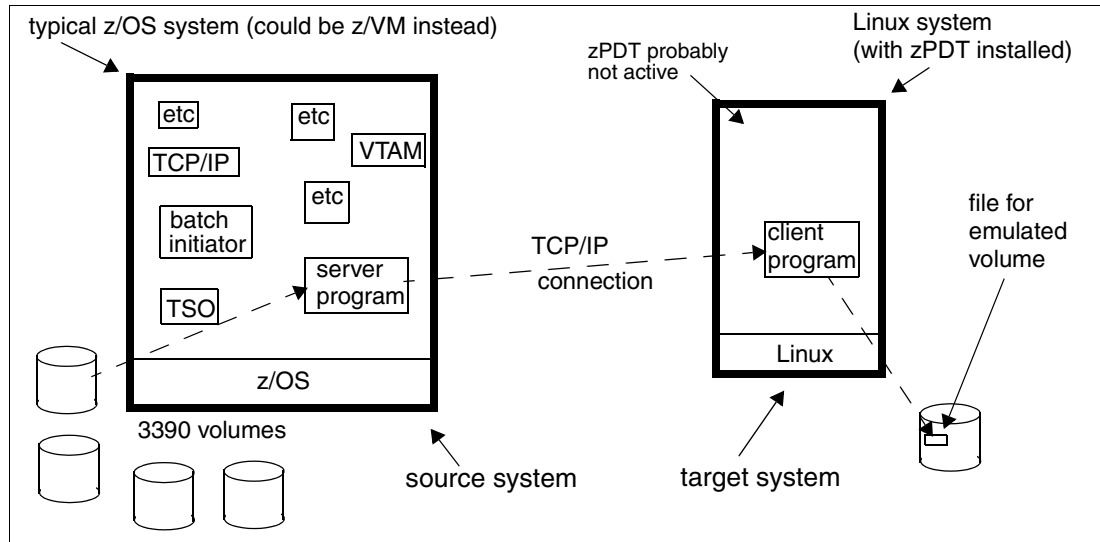


Figure 11-1 Volume migration overview (z/OS version)

## 11.1 Warnings

There are limitations on simply copying and using volumes from a z/OS system (and also from a z/VM system, with slightly different details). These limitations are not related to zPDT or to the migration utility described in this chapter. A z/OS disk volume is not necessarily a stand-alone entity, depending on what is on the volume. In particular, if the volume contains VSAM data sets of any type (including catalogs) then the volume contents are usable only if the complete VSAM metadata is available. Volume AAA may contain a VSAM data set that is cataloged on volume BBB. The VSAM information on volume AAA (including metadata in the VVDS) must be synchronized with the catalog information on volume BBB. In this simple example, migrating (copying) both volumes might suffice, provided the catalog on volume BBB is later properly connected to the master catalog on the target system.

In general, volumes that do not contain VSAM data sets (or VVDS material) are safer to migrate in a stand-alone fashion. Migrating all the volumes of a z/OS system should be safe because this would include all the relevant catalogs and VSAM data volumes.

A second warning is related to the migration utility described in this chapter. No enqueue functions are used by this utility. Transferring an active volume is probably not a good idea. If the volume is active, it might have logical errors when the transferred copy is used (for example, the VTOC might not reflect an additional extent that was allocated after the tracks containing the VTOC were sent or the contents of a z/VM minidisk on the volume might be changing as the associated tracks are copied).

With a z/VM system, the VM directory must be synchronized with any DASD volumes containing minidisks.

## 11.2 Operational characteristics of the migration utility

The following characteristics of this utility are important:

Complete volumes (3380 or 3390) are transferred. This includes IPL text, volume labels, VTOC, and *unallocated space*. The *logical* contents of the volume are not examined. Data sets on the volume are not recognized. The utility simply copies and transfers all the tracks on the volume. It does not check whether the tracks are allocated (VTOC or VM equivalents) or in use (ENQ) or linked to a specific catalog (for VSAM, for example).

- ▶ A **read track** CCW is used to read the data on each track of a CKD volume. The amount of data on each track is variable. This means the time to transmit a volume is variable, depending on how much data is on each track.
- ▶ Track data is not compressed for transmission.
- ▶ The source z/OS or z/VM (where the server component runs) is typically a large System z, but this is not required. It could be a zPDT system. The receiving Linux side must be a Linux system with zPDT installed (but probably not active). The received copy of the source disk volume is stored in the awsckd (or awsfba or awstape for z/VM) format used by zPDT.
- ▶ Specific RACF definitions are required for the source z/OS side. These definitions can protect the utility from misuse.
- ▶ The utility server program must reside in an authorized library for z/OS.
- ▶ The utility server program is typically started as a batch job. It automatically terminates after 15 minutes of inactivity.
- ▶ The utility client program is run as a normal Linux command. It is possible to create a script file with multiple transfer commands, so that multiple volumes may be transferred in an unattended manner.
- ▶ TCP/IP port 3990 is used by default. The port number may be changed when starting the server and client.
- ▶ Both 3380 and 3390 volumes, any size, may be migrated. This includes 3390 EAVs (“large volumes”).
- ▶ In practice, this utility is likely to run unattended because copying multiple volumes can take considerable time. We suggest you first try a single volume and monitor the operation while it is running.
- ▶ A list of volumes to be migrated may be created by simply creating a suitable Linux file with a command for each volume.
- ▶ Only one transfer may be active for the server. It is possible to run multiple servers in parallel (with different IP port numbers for each) but the usefulness of this is questionable.
- ▶ The z/OS version is only for the migration of CKD DASD volumes. The z/VM version can migrate CKD and FBA DASD volumes and tape volumes.

## 11.3 Installation of the migration utility for z/OS

A number of steps are required to install the migration utility. They are:

1. Upload the server module and install it in an authorized z/OS library. (The server module is provided in the `/usr/z1090/bin` directory that contains all the zPDT executables. It is an unloaded PDS member that has been processed by the TSO XMIT command.)
2. Provide the required RACF definitions.
3. Determine whether TCP/IP port 3990 (on the z/OS side and the Linux side) has been assigned for other purposes.
4. Create a batch job to run the server.

### 11.3.1 Server installation

File /usr/z1090/bin/ZOSSERV.XMIT must first be uploaded to a z/OS data set with DCB characteristics RECFM=FB, LRECL=80, BLKSIZE=3120. This XMIT file contains an unloaded PDS load library with one member. You can begin restoring this material by preallocating two data sets with the following job:

```
//OGDEN77 JOB 1,BILL,MSGCLASS=X
// EXEC PGM=IEFBR14
//A DD DISP=(NEW,CATLG),UNIT=3390,VOL=SER=ZBSYS1,SPACE=(TRK,5),
//   DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120,DSORG=PS),
//   DSN=IBMUSER.SEQ.HOLDING
//B DD DISP=(NEW,CATLG),UNIT=3390,VOL=SER=ZBSYS1,SPACE=(TRK,(3,3,3)),
//   DCB=(LRECL=0,BLKSIZE=32760,RECFM=U),DSN=IBMUSER.PDS.HOLDING
```

(The volser and DSNs are arbitrary. An experienced z/OS system programmer can handle this upload and installation in multiple ways. We present a basic example here. You can, for example, use ISPF to create these two holding data sets.)

After the receiving data set is ready, use binary mode and either IND\$FILE or FTP to transfer /usr/z1090/bin/ZOSSERV.XMIT to IBMUSER.SEQ.HOLDING. It is a relatively small file. *Be certain to use a binary transfer.*

After the module has been uploaded to the sequential holding data set, issue the following TSO commands (using the appropriate data set names, of course):

```
RECEIVE INDATASET('IBMUSER.SEQ.HOLDING')
INMR901I DATASET SHANNON.CKDTOPEN.....
INMR906A Enter restore parameter or 'DELETE' or 'END' +
DATASET('IBMUSER.PDS.HOLDING')
INMR001I Restore successful ....
```

The name of the executable module, once restored, is ZPDTMSRV. You must select an authorized library (preferably on the LNKLIST) to contain the server module. You must have authority to update this library. We use USER.LINKLIB as an example of an authorized library in the LNKLIST, but your source system is probably different. Userid IBMUSER typically has update authority for all system libraries and we use this userid in our example. Again, your source system may be different.

You need to copy member ZPDTMSRV from your temporary PDS to your authorized library. The easiest way to do this is by using ISPF option 3.3. In the first panel (of ISPF 3.3), select:

```
Option=====> c
....
("from" data set name)
Name. . . . . 'IBMUSER.PDS.HOLDING'
      (press Enter)
("to" data set name)
Name. . . . . 'USER.LINKLIB'
      (press Enter)
. ZPDTMSRV                (overtpe the period with the letter S and Enter)
```

This completes the server installation. The holding data sets can be deleted.

## 11.3.2 RACF requirements

**Important:** Consult with the RACF administrator for your source z/OS system before making any RACF changes. This is especially important if the DASDVOL class is active in the installation.

The server program *requires* RACF class DASDVOL to be active and the server program must have at least READ access to all volsers to be transferred. You must determine whether the DASDVOL class is active and used on the system where you install the migration utility server. You can do this by logging onto TSO with a userid having RACF SPECIAL authority and issuing these TSO commands from a READY prompt or ISPF option 6:

```
SETROPTS LIST                (check the list of active classes)
RLIST DASDVOL *              (see if any profiles are defined)
```

If these checks are negative, you can assume that the DASDVOL class is not being used.

### DASDVOL not already in use

The next step is to decide whether only certain volumes should be subject to use by this migration utility or whether all volumes might be accessed for migration. If you elect to potentially allow migration of all volumes, enter these TSO commands:

```
SETROPTS CLASSACT(DASDVOL)    (activate the DASDVOL class)
SETROPTS RACLIST(DASDVOL)     (optional, but recommended here)
RDEFINE DASDVOL ** UACC(ALTER) (allow universal alter access)
SETROPTS RACLIST(DASDVOL) REFRESH (if you RACLISTed the class)
```

Be aware that the DASDVOL class is used only by a selected group of utility programs, such as dump/restore. Allowing UACC(ALTER) does not open all your data sets to access by all users. Whatever RACF data set protection you have in place (via ADDSD and PERMIT commands) is still effective.

This is all the RACF setup required if DASDVOL was not initially active and if you want to allow the migration server to access any volume.

If you want to limit the volumes subject to migration, you should work with an experienced RACF administrator. The general technique is to restrict global access to DASDVOL (perhaps with a UACC(NONE) condition) and then issue PERMIT commands to cover the volumes you want to migrate. For example:

```
PERMIT VOL123 CLASS(DASDVOL) ID(IBMUSER) UACC(READ)
PERMIT ADCD* CLASS(DASDVOL) ID(IBMUSER) UACC(READ)
SETROPTS RACLIST(DASDVOL)REFRESH          (if you RACLISTed DASDVOL)
```

In this example, volser VOL123 and any volser beginning with ADCD can be accessed by the migration utility.

The use of the DASDVOL class may have side effects on other utility programs. If DASDVOL was not active before your migration activities, you may want to deactivate it when the migration activities are completed:

```
SETROPTS NORACLIST(DASDVOL)
SETROPTS NOCLASSACT(DASDVOL)
```

## DASDVOL already in use

If you find that the DASDVOL class is active on your source system, we *strongly* suggest that you discuss the situation with the system programmers managing the source system. Your requirements are simple: you (meaning the userid who will run the migration server program) need DASDVOL READ access to whatever volsers you intend to migrate.

## TCP/IP port

By default, the migration programs use TCP/IP port 3990, but this can be changed. You can look at the TCP/IP PROFILE on the z/OS system to see whether port 3990 is reserved for another application. However, another application could dynamically acquire the port. There is no easy way to prevent this. We suggest specifying a different port number only if there are error messages when you try to start the migration server or client.

## 11.4 Operation of the server under z/OS

The server should not be started until you are ready to use it. It automatically terminates after ten minutes of inactivity. The following JCL could be used to start the server:

```
//MIGSERV JOB 1,OGDEN,MSGCLASS=X,TIME=1440
//ZPDTMIG EXEC PGM=ZPDTMSRV,REGION=0M,PARM='3990'
//SYSUDUMP DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSTCPD DD DISP=SHR,DSN=ADCD.Z111.TCPPARM(TCPDATA)
```

Notes:

- ▶ The PARM field is not needed if you use the default port number (which is 3990).
- ▶ The TIME=1440 parameter is recommended because transmitting a full volume (or several volumes) can take considerable time.
- ▶ The SYSTCPD statement must point to the DATA used by the z/OS TCP/IP stack. You must determine the correct data set name for your z/OS system. One way to do this is to examine the procedure used to start TCP/IP on your system.
- ▶ If your authorized library is not in LNKLST, you will need a JOBLIB or STEPLIB statement.
- ▶ The user submitting this job must have a userid that has at least READ access to the appropriate volsers in DASDVOL.

This job must be submitted by a userid who has at least READ access to the required volumes in the RACF DASDVOL classhhh. If a transfer is interrupted, a subsequent command to transfer the same volume will restart with the last successful cylinder that was transferred.

## 11.5 Installation of the server under z/VM

The z/VM system must be Release 5.4 or later.

The ZVMSERV.XMIT file is located with other zPDT binary material in /usr/z1090/bin on the zPDT system. This must be uploaded in BINARY, FIXED LRECL 80 format to a CMS user's A disk. (This file essentially contains a card deck.) After this is complete, do the following from CMS:

1. Issue the command CP SPOOL PUN.
2. Issue the command PUNCH ZVMSERV XMIT A (NOH) << I closed the paren. ok? >>.

3. Issue the command RDRLIST.
4. Beside the file in your RDRLIST, issue the command RECEIVE.

This should result in the reconstructed executable file on your CMS A disk. This completes the server installation.

## 11.6 Operation of server under z/VM

Your z/VM must have TCP/IP active and connected to the appropriate network. The CMS user running the migration utility must have access to the volumes to be migrated.

The CMS user starts the utility by issuing a ZVMSERV command; a single operand specifies the TCP/IP port number and this defaults to port 3990. When started, the server should indicate that it is waiting for a client connection. The server will time out after 10 minutes without a client connection.

If a z/VM mini-disk is migrated, it will appear as a small 3390 volume on the receiving system. That is, it will no longer be a mini-disk on a larger z/VM volume.

## 11.7 The client commands

There are three client commands:

- ▶ **hckd2ckd** - used with both z/OS and z/VM to migrate a CKD DASD volume.
- ▶ **hfba2fba** - used only with z/VM to migrate a FBA DASD volume.
- ▶ **htape2tape** - used only with z/VM to migrate a tape volume to a zPDT awstape volume.

This general syntax of the client commands (entered on the Linux client machine, using a normal Linux command window) is:

```

hxxx2xxx host[:port] outfile [-n          ] [-v          xxxxxx] [-u          aaaa]
                               [--norestart] [--volser xxxxxx] [--unit aaaa]

                               [-e eof-count] [-n]
                               [--eof eof-count]

```

`host` is the TCP/IP name of the system with the matching server program. This may be a dotted-decimal address or a name that can be resolved by Linux TCP/IP.

`:port` is a TCP/IP port number to be used by both the client and server program. It defaults to 3990.

`outfile` is a file name (on the current Linux) system where the migrated volume is placed (in `awsckd`, `awsfba`, or `awstape` format).

`-n` or `--norestart` indicates that a previous incomplete volume transmission is not to be restarted where it ended; the complete indicated volume is to be sent again. This applies only to `hckd2ckd`.

`-n` (for `htape2tape`) means the `awstape` output is not to be compressed.

`-v` or `--volser` indicates the 3380/3390 volume (on the remote z/OS system) that is to be copied (migrated).

`-u` or `--unit` indicates the address (device number) of the volume that is to be copied (migrated).

**-e** or **--eof** indicates the number of consecutive tape marks that will indicate the end of the input tape. This is used only with z/VM tapes. The default is two tapemarks.

Either the **-u** or **-v** parameter must be supplied for DASD, but not both; the **-u** parameter would normally be used for tapes.

Once the server is started on the z/OS or z/VM system, the client may be started on the Linux system. Remember that zPDT need not be operational for this. (We generally recommend that it should not be operational, because the migration utility can place a heavy load on the LAN interface.) Examples of commands that could be used to run the client are as follows:

```
$ hckd2ckd 192.168.0.99 /z/VOL123 -v VOL123
$ hckd2ckd BIG.ZOS.ADDR:4990 /z/VOL678 -u A8F
$ hckd2ckd 192.168.0.99:4990 /z/host.WORK23 -v WORK23 --norestart
```

The first operand is the IP address of the z/OS system where the server is running. This may be in dotted-decimal form or as a name that the Linux system can resolve. The TCP/IP port number can be changed as shown in the examples, where we use port 4990 in two cases. (The server must have been started using the same port number, of course.) The second operand is the Linux file name used to store the migrated volume. Either the **-v** or **-u** parameter must be specified. The **-v** parameter is a volser and the **-u** parameter is a device address (device number) on the server system; these determine which volume is to be processed.

## 11.8 Additional notes

The migration of a tape volume results in a compressed awstape output file. The **-n** option will produce an uncompressed awstape output file.

### Restart

The client program maintains state information on the Linux system and can restart a transfer if the link fails (or if either system fails for some reason). You can prevent an automatic restart with the indicated parameter (which means that the volume copy will start from the beginning again). The restart function applies only to CKD volumes. It is not available for FBA or tape volumes.

### Devmap

After a volume has been migrated to Linux, you can add it to your devmap and access it from zPDT. You should also check the permission bits for the file. (The 1090 must have read/write access to it.) Our examples are in terms of z/OS volumes, but the volume could be for z/VM, z/VSE, or Linux for System z.

### Labelled tapes

The migration utility (z/VM version) does not inspect tape labels; they are simply treated as files. By default, the migration function stops when two adjacent tape marks are encountered. This can have two side effects:

- ▶ A null file in a labelled multifile volume can produce two adjacent tape marks that do not indicate the end of the tape. You must manually handle this situation. (Imbedded null files on tape are considered bad practice; this situation should be rare.)
- ▶ A multivolume labelled tape data set has only a single tape mark at the end of the first volume(s) << can there be more than one first? there is only one last...>>; the last volume is terminated by two tape marks. You can handle this by using the default termination

indicator (two tape marks). This will produce an error message at the end of each of the initial volume(s); the error can be ignored.

## Linux volumes

We recently discovered an interesting situation when migrating Linux for System z volumes. Our particular experience was with SLES-10 SP1 (for System z), but it may apply to other distributions.

When installing this distribution there are a number of *fstab options* that can be selected for controlling disk volume mounts. These include mounting by device name, volume label, UUID, device ID, or device path. The default is to mount by device ID. This produces a boot parameter list (and fstab) something like this:

```
parameters='root=/dev/disk/by-id/ccw-IBM.750000000M1881.2c23.1c-part1'
```

This disk identification is unique to the original disk drive and is useless when the volume is copied or migrated to another disk. In this situation, the migrated Linux volumes could not be booted. This identification is best changed when installing Linux by selecting the use of a volume label (for example, LABEL=rootfs) or device name (for example, /dev/dadda1) when initially creating the disk partitions. The naming can be changed later by carefully editing /etc/zip1.conf and /etc/fstab. In any event, the naming should be changed before migrating the volumes.

## Multiple TCP/IP stacks

A z/OS system with multiple TCP/IP stacks presents an additional complication. In this situation, an additional step is needed in the server job:

```
//MIGSERV JOB 1,OGDEN,MSGCLASS=X,TIME=1440
//STEPO EXEC PGM=BPXTCAFF,PARM='TCP342'
//*
//ZPDTMIG EXEC PGM=ZPDTMSRV,REGION=0M,PARM='3990'
//SYSUDUMP DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSTCPD DD DISP=SHR,DSN=ADCD.Z111.TCPPARM(TCPDATA)
```

The BPXTCAFF program is used to associate a specific TCP/IP stack with the current address space. The PARM value is the job name used to start the desired TCP/IP stack.

## Typical client usage

A typical use of the client might be as follows:

```
$ hckd2ckd 192.168.0.81 /z/ZBRES1 -v ZBRES1
AWSHTC015W No port specified, defaulted to 3990
AWSHTC090I Host name : 192.168.0.81:3990
AWSHTC091I Restart : No
AWSHTC095I Vol-Ser : ZBRES1
AWSHTC096I Output : /z/ZBRES1
AWSHTC097I Transferring 3990 volume of 3339 cylinders
AWSHTC098I Cylinder nnnn ...
```

We found that transferring a fairly full 3390-3 volume on a private 100 Mbps LAN took approximately 11 minutes. This consumed roughly 150 processor seconds on the source z/OS system (which was a zPDT system on a moderate-performance PC).

When the migration is complete, the cylinder number displayed will be one less than the actual number of cylinders transferred. Also, there may be a delay (perhaps up to 30

seconds) between the last message and the time the command ends. Both these conditions are normal.

### **Migrating a list of volumes**

Using **gedit** or another editor, you could create a Linux file named, for example, mig:

(contents of the mig file)

```
hckd2ckd 192.168.0.81 /z/ZOS111/ZBRES1 -v ZBRES1
hckd2ckd 192.168.9.01 /z/ZOS111/ZBRES2 -v ZBRES2
hckd2ckd 192.168.9.01 /z/ZOS111/ZBSYS1 -v ZBSYS1
hckd2ckd 192.168.9.01 /z/ZOS111/ZBUSS1 -v ZBUSS1
hckd2ckd 192.168.9.01 /z/ZOS111/backup/MYVOL1xx -v MYVOL1
```

You could start the server program on the z/OS host and then execute the commands in your mig file:

```
$ ./mig (execute the commands in file named mig)
```

This will transfer all the volumes listed without any further manual intervention.

## Channel-to-channel

The `awsctc` device manager provides channel-to-channel (CTC) functions using TCP/IP communication paths. As shown in Figure 12-1, the connection can be within the same zPDT instance, between zPDT instances on the same machine, or between zPDT instances on different machines. Among other functions, CTC can be used by z/OS for GRS “rings”, NJE connections, TCP/IP connections, and so forth.

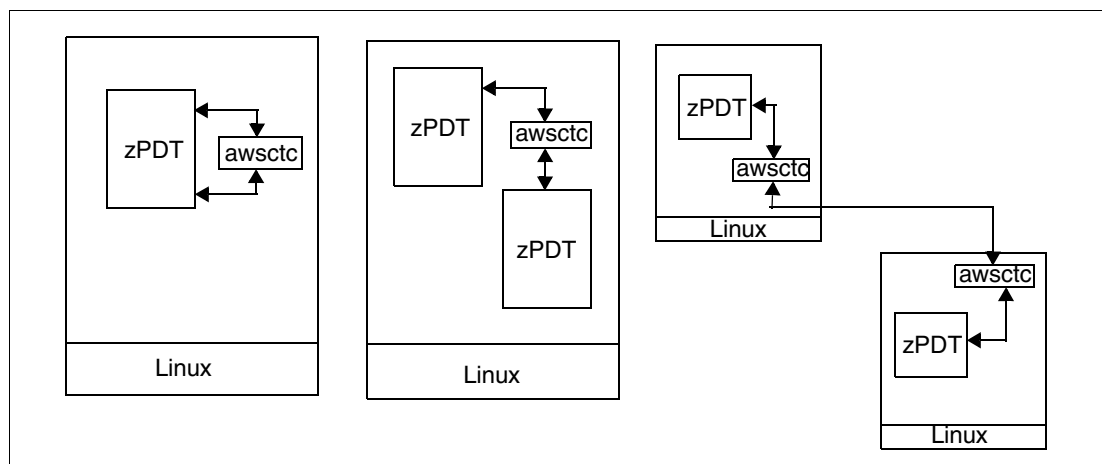


Figure 12-1 CTC links for zPDT

The `awsctc` device manager emulates IBM 3088-4 (or 3088-8) control units and devices.<sup>1</sup> The 3088 is an older product that provided the “middle” function for connecting two channels to each other. Each channel saw the 3088 as a control unit that provided a number of devices (each of which was a connection path). Modern systems have this control unit function integrated with the channels, and physical 3088 boxes are no longer used. However, the logical function has not changed.

<sup>1</sup> These are parallel channel control units. ESCON CTC operation is not emulated.

The devmap stanza for awsctc appears as follows:

```
[manager]
name awsctc 75
device E40 3088 3088 ctc://otherhost:3088/E42
device E41 3088 3088 ctc://192.168.0.70:3088/E43
```

The last parameter of the device statement contains three elements:

- ▶ A TCP/IP address (in dotted decimal form or as a name that can be resolved by Linux)
- ▶ Following a colon, a TCP/IP port number that is to be used on both the local and other system. The same port number is used on both ends of the connection.<sup>2</sup>
- ▶ Following a slash, the device number (address) of the corresponding CTC device on the remote system. To specify this you must know how the devmap is defined on the other system.

An example of a simple connection between two zPDT instances in two different machines is shown in Figure 12-2.

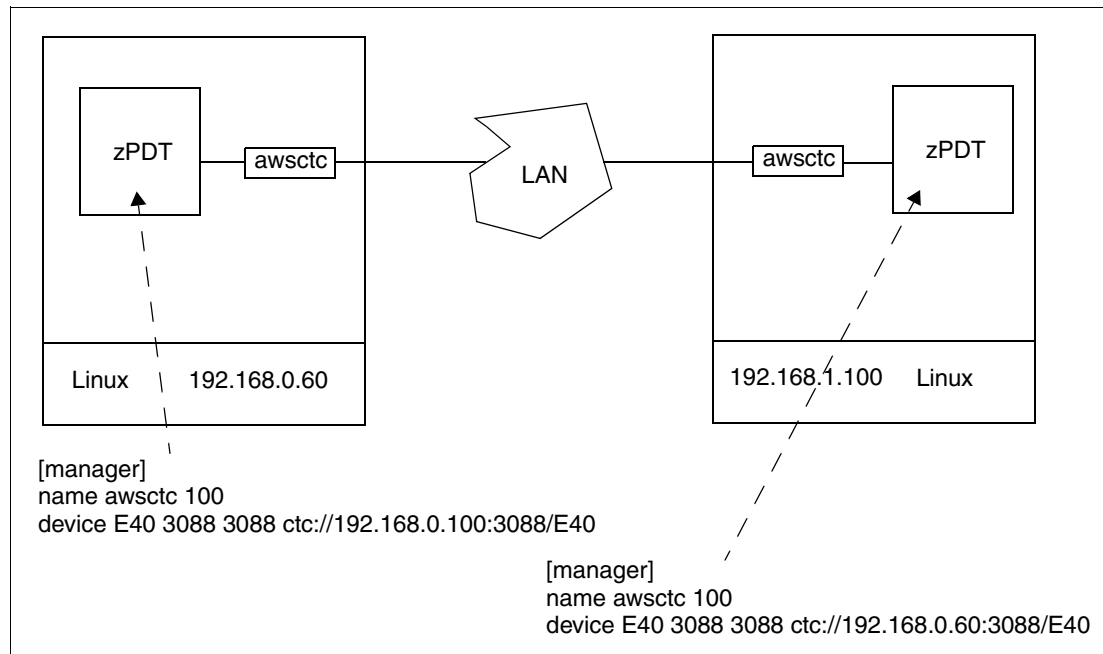


Figure 12-2 Simple two-system CTC definition

In this example, two separate zPDT instances are shown. Because they are separate instances, the CUNUMBR (100 in the example) can be the same in both definitions (but there is no requirement to do this, of course). Both instances elected to use the same device number (E40) for the CTC device, although there is no requirement to do this. Both ends of the connection *must* specify the same port number (3088 in the example). Each definition specifies the IP address of the other base Linux system.

### Devmap notes

The "ctc://" is optional, but if a protocol is specified it must be "ctc". "Well known" port numbers below 1024 cannot be used; the port number must be between 1024 and 65535. The IP address string cannot be specified via an **awsmount** command.

<sup>2</sup> We use port 3088 for these examples because it is easy to remember. There is nothing special about this port number.

Each CTC address must be defined. A real 3088 has multiple device numbers defined in blocks of 8, 16, 32, or 64 devices. For example, a 3088-4 would have 16 devices starting at a specified address. The emulated CTC does not do this. Each specific device number must be defined.

A CTC connection within the same zPDT instance might be defined as follows:

```
[manager]
name awsctc 300
device E40 3088 3088 ctc://localhost:3088/E42
device E42 3088 3088 ctc://localhost:3088/E40
```

Note that the two device statements refer to each other's device number.

### Status display

The AWSSTAT command may be used to display the status of the CTC device. When the device's peer is not yet available, the status flip flops between *Connecting* and *Accepting*. Once the peer is available, the status changes to either *Connected* or *Accepted*. The *accepted* side is considered the A side for protocol conflict resolution. This generally does not make any difference to the user. Once data begins to flow, the Accepted/Connected is shortened to A or C and the number of send/receive bytes is displayed.

## 12.1 z/OS usage example

Configurations using CTC can be complex. We have taken a basic NJE example, as shown in Figure 12-3.

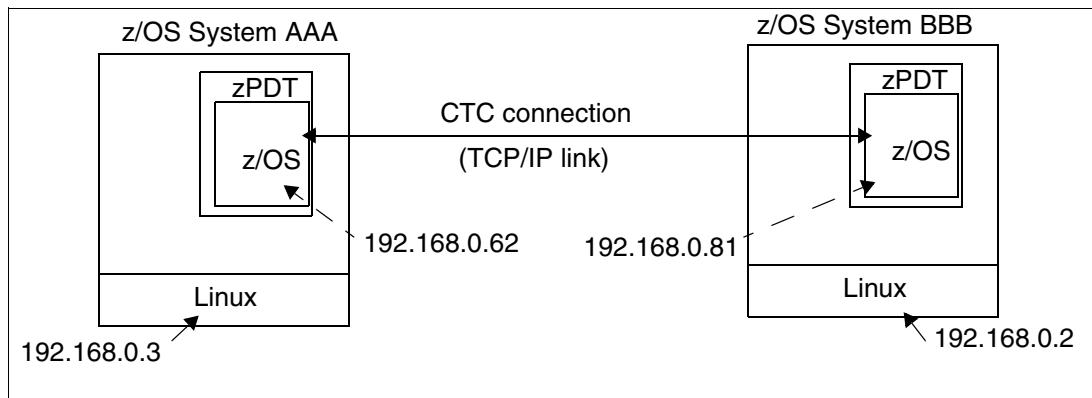


Figure 12-3 Trivial NJE setup

The JES2PARM data for system AAA might include the following:

```
NJEDEF DELAY=360,LINENUM=2,JRNUM=2,JTNUM=2,NODENUM=2,OWNNODE=1,
        PATH=1,RESTTOL=100,RESTMAX=8000000,RESTNODE=100,SRNUM=2,
        STNUM=2,MAILMSG=YES,TIMETOL=1440
LINE(1) UNIT=E40,TRANSPAR=YES
NODE(1) NAME=AAA,PATHMGR=NO
NODE(2) NAME=BBB,PATHMGR=NO
CONNECT NODEA=AAA,NODEB=BBB
```

The JESPARM data for system BBB might include the following:

```
NJEDEF DELAY=360,LINENUM=2,JRNUM=2,JTNUM=2,NODENUM=2,OWNNODE=2,
```

```

        PATH=1,RESTTOL=100,RESTMAX=8000000,RESTNODE=100,SRNUM=2,
        STNUM=2,MAILMSG=YES,TIMETOL=1440
LINE1 UNIT=E40,TRANSPAR=YES
NODE(1) NAME=AAA,PATHMGR=NO
NODE(2) NAME=BBB,PATHMGR=NO
CONNECT NODEA=AAA,NODEB=BBB

```

These examples assume that device number E40 is defined in z/OS as a CTC device. (This is true for current z/OS AD-CD systems.) Be careful to place the IP address of the remote Linux system (and not the remote z/OS system) in the devmaps.

The devmap for system AAA might contain the following stanza:

```

[manager]
name awsctc 300
device E40 3088 3088 ctc://192.168.0.2:3088/E40

```

The devmap for system BBB might contain the following stanza:

```

[manager]
name awsctc 300
device E40 3088 3088 ctc://192.168.0.3:3088/E40

```

The status of the CTC connection can be monitored with the **awsstat** command:

```

$ awsstat E40 (use the correct device number, of course)

```

The device status usually starts as *connecting* or *accepting*, and later switches to *connected* or *accepted*. This may take some time. After the connection is accepted and used, the status displays byte counts, such as C-160/162.

???We found it best to wait until the CTC link is established before IPLing z/OS. The link can be checked with the **awsstat E40** command (using the correct device number for your CTC, of course). This normally takes only a few seconds, but can take longer in rare cases. We then needed to issue commands on each JES2 system to make the connection operational:

```

$$ N,LINE1 (sometimes needed)
$$ LINE1

```

A job submitted on system AAA could be directed to system BBB for execution by the following JCL:

```

//MYJOB JOB 1,OGDEN,MSGCLASS=X,USER=IBMUSER
// XMIT DEST=BBB
//MYJOB2 JOB 1,OGDEN,MSGCLASS=X,USER=IBMUSER
//STEP1 EXEC PGM=IEFBRI4

```

Notice the two JOB statements in this example. The first is needed to “introduce” the XMIT statement into the job stream. The XMIT statement sends everything after it to the indicated node, including the second JOB statement. The first JOB statement (before the XMIT) is not sent to the remote node.

The following JES2 commands can be useful when working with this NJE connection:

```

$DCONNECT (very useful status display)
$$ LINE(1)
$$ N,LINE1 (discover dynamic connections)
$D NODE(*)
$D LINE
$D DESTID(*)

```

```

$D PATH(*)
$P LINE(1)                (stop line)
$E LINE(1)                (reset line)
$D NJEDEF                 (display NJE definitions)
$N,D=BBB,'$D NJEDEF'     (send JES2 command to other node)

```

A JES2 cold start may be necessary to enable NJE changes to JES2PARM. You need to stop the line, **\$P LINE(1)**, when stopping JES2. If the \$P is not effective, try **\$E LINE(1)** followed by **\$P LINE(1)**..

## 12.2 Multiple instances and z/VM

The following material outlines several uses of CTC connections with two instances of z/VM.<sup>3</sup> This material is intended as a general overview and does not provide complete step-by-step instructions for implementation and usage. It is very unlikely that all the links shown in this example would be present in any practical system.

### 12.2.1 Devmaps

Three devmaps are needed for these examples. One is for a group controller (for shared devices) and two are for the z/VM instances. Many of the choices are arbitrary.

#### **Group controller (ibmgroup)**

```

[system]
members ibmsys1 ibmsys2
3270port 3270

[adjunct-processors]    # not needed for the CTC examples
crypto 1                # included to illustrate shared crypto
crypto 2

[manager]
name aws3274 0002
device 0020 3279 3274
device 0021 3279 3274    # more devices could be included

[manager]
name awsosa 0009 --path=A0 --pathtype=OSD --tunnel_intf=y
device 0E00 osa osa
device 0E01 osa osa
device 0E02 osa osa

```

#### **ibmsys1 instance**

```

[system]
memory 1024m
processors 1
group ibmgroup

[adjunct-processors]    #not needed for CTC; included as an example
domain 0 1
domain 1 1

```

<sup>3</sup> Thanks to Bruce Hayden, of the Washington Systems Center, for this material.

```
[manager]
name awsckd 0001
device 1000 3390 3990 /z1/540res      #unshared disks
device 1001 3390 3990 /z1/540sp1
device 1002 3390 3990 /z1/540pag
device 1003 3390 3990 /z1/540w01
device 1004 3390 3990 /z1/540w02
```

```
[manager]
name awsctc 0075
device 700 3088 3088 ctc://localhost:3700/700  #for TCPIP
device 700 3088 3088 ctc://localhost:3701/701  #for TCPIP
device 710 3088 3088 ctc://localhost:3710/710  #for TSAF
device 720 3088 3088 ctc://localhost:3720/720  #for ISLINK
device 740 3088 3088 ctc://localhost:3740/740  #for RSCS
```

### ***ibmsys2 instance***

```
[system]
memory 1024m
processors 1
group ibmgroup
```

```
[adjunct-processors]      #not needed for CTC; included as an example
domain 3 2
domain 4 2
```

```
[manager]
name awsckd 0001
device 1000 3390 3990 /z2/540res      #unshared disks
device 1001 3390 3990 /z2/540sp1
device 1002 3390 3990 /z2/540pag
device 1003 3390 3990 /z2/540w01
device 1004 3390 3990 /z2/540w02
```

```
[manager]
name awsctc 0075
device 700 3088 3088 ctc://localhost:3700/700  #for TCPIP
device 700 3088 3088 ctc://localhost:3701/701  #for TCPIP
device 710 3088 3088 ctc://localhost:3710/710  #for TSAF
device 720 3088 3088 ctc://localhost:3720/720  #for ISLINK
device 740 3088 3088 ctc://localhost:3740/740  #for RSCS
```

### **Description**

Notice that each instance has its own copy of the z/VM disks; they are in separate directories, /z1 and /z2.

### ***ISLINK***

An ISLINK creates an ISFC (Inter-System Facility for Communications) connection. It is the easiest link because it involves only CP commands and does not require a virtual machine or userid. The VM system identifiers of the two connected systems must be different. Using the devmaps shown above, the following commands (issued by MAINT, for example) activate the link:

```
CP ACTIVATE ISLINK 0720          (command on ibmsys1)
```

```
CP ACTIVATE ISLINK 0720          (command on ibmsys2)
```

This should result in the message HCPALN2702I Link 0720 came up. The command Q ISLINK can be used to display the status of the connection.

### **TSAF**

TSAF is an older function and probably would not be used if IFSC is available. To try it, use commands such as the following (issued by MAINT on both systems):

```
CP XAUTOLOG TSAFVM
CP ATT 0710 TSAFVM
```

Then log onto TSAFVM and enter ADD LINK 0710 on both sides. The status of the links can be displayed with Q LINKS ALL.

### **RSCS**

RSCS can be more complicated. The RSCS product is not enabled by default. You can check this with the command Q PRODUCT STATE ENABLED. If RSCS is not in the list, it can be enabled (using MAINT) by the command SERVICE RSCS ENABLE followed by PUT2PROD.

A configuration file is needed on each system. The configuration files are placed on the RSCS 191 disk with the name RSCSTCP CONFIG. An example for each system might be as follows:

```
(for ibmsys1)
LOCAL IBMSYS1      * RSCS
LINKDEFINE IBMSYS2 TYPE NJE LINE 740 QUEUE PRI NODE IBMSYS2
PARM IBMSYS2  STREAMS=2 MAXU=2 MAXD=10 LISTPROC=NO TA=1 TAPARM='TH=100'

AUTH * OPERATOR * CP
AUTH * MAINT    * CP

(for ibmsys2)
LOCAL IBMSYS2      * RSCS
LINKDEFINE IBMSYS1 TYPE NJE LINE 740 QUEUE PRI NODE IBMSYS1
PARM IBMSYS1  STREAMS=2 MAXU=2 MAXD=10 LISTPROC=NO TA=1 TAPARM='TH=100'

AUTH * OPERATOR * CP
AUTH * MAINT    * CP
```

After the configuration files are available, start RSCS with the command XAUTOLOG GCS (issued on both systems). After RSCS starts issue the command SMSG RSCS START IBMSYS1 on ibmsys2 and SMSG RSCS START IBMSYS2 on ibmsys1. The status of the RSCS connection can be displayed with SMSG RSCS Q SY.

### **TCP/IP**

TCP/IP requires a pair of CTC links, one for read and one for write. The easiest way to set up TCP/IP for z/VM is with the IPWIZARD. With this you must assign hostnames and domain names; in an isolated environment these can be arbitrary names. For an isolated environment with only two nodes the gateway address does not matter; you might use 10.1.1.1.<sup>4</sup> Use CTC0 as the interface name and address 0700 (from our sample devmap). We assigned IP address 10.1.1.2 to ibmsys1 and 10.1.1.3 to ibmsys2, and used a mask of 255.255.255.0. The configuration dialog includes positional parameters to indicate which device to use as the

<sup>4</sup> Remember that the default address of the tunnel connection to the base Linux is 10.1.1.1. This default is used in the sample devmap.

read channel and which to use as the write channel; this parameter could be 0 on ibmsys1 and 1 on ibmsys2. The status of TCPIP can be checked by:

```
VMLINK TCPMAINT 592  
NETSTAT DEVL
```

You can force TCPIP to restart with the following:

```
FORCE TCPIP  
XAUTOLOG TCPIP
```



## Cryptographic adapter

A 1090 system (starting with the spring 2010 release) can emulate several cryptographic adapters as CEX2C devices. Each of the CEX2C emulated adapters runs as separate Linux processes and, if sufficient base processors are available to permit these threads to be dispatched in parallel by Linux, can run asynchronously with the 1090 CPs.

Do not confuse a cryptographic adapter with the cryptographic instructions that are always available with a 1090 system. These are the KM, KMC, KIMD, KLMD, and KMAC instructions that provide a number of fundamental cryptographic operations. Also, note that the terms *cryptographic adapter* and *cryptographic coprocessor* are used synonymously in this document.

The cryptographic instructions may be coded directly in a program or used through ICSF programming interfaces. For practical purposes, the cryptographic coprocessor facilities are available only through ICSF programming interfaces.

TKE systems, which are personal computers with unique software and an appropriate cryptographic adapter, are not used with the 1090."

## 13.1 Background information

A cryptographic coprocessor is represented by an adjunct-processor (AP) process in the 1090. A System z CP sends and receives work entries through coprocessor queues that are known as *domains*.

The *accelerator mode* of a cryptographic coprocessor is not provided for the 1090.

Each cryptographic coprocessor has 16 domains; each z/OS instance uses a different domain (this is specified in the ICSF parameters). If multiple coprocessors are defined, then z/OS uses the same domain in each coprocessor. If multiple coprocessors are defined, z/OS can dispatch requests to all of them (using the same domain number in each processor).

Within the 1090, the cryptographic coprocessor keys (and other state information) are stored in the `~/z1090/srdis` directory. There is a separate subdirectory for each defined coprocessor. The subdirectory name is simply the coprocessor number. For example, `~/z1090/srdis/5` would contain the data for coprocessor number 5. These directories are created automatically by zPDT.

Any tampering with this information can produce unpredictable results. However, a complete coprocessor subdirectory may be deleted as a way of reinitializing (zeroing) that coprocessor. (However, the `ap_zeroize` command, described later, is the recommended way to reinitialize a coprocessor.)

It is important to understand that zPDT cryptographic functions are intended for development purposes and not for security. While the format of the key data in `~/z1090/srdis` is not documented, it is certainly not secure in any cryptographic sense.

## 13.2 Devmap specification

The zPDT specification for cryptographic adapters is part of the devmap and consists only of a simple *adjunct-processors* stanza:

```
[system]
memory 4000m
3270port 3270
processors 1

[adjunct-processors]
crypto 0
crypto 1           #(more than one cryptographic adapter is possible)

[manager]
name .....
```

The zPDT architecture allows up to 16 or 64 cryptographic coprocessors, depending on the overall configuration.<sup>1</sup> z/OS allows a maximum of 16.

---

<sup>1</sup> A zPDT group controller instance may have up to 64 coprocessors defined, otherwise the limit is 16.

## 13.3 Initial ICSF startup

For practical purposes, the ICSF software functions are needed to initialize cryptographic adapters. Following is a brief outline of the steps we took to customize and use the ICSF panels on a z/OS 1.11 system (which was the AD-CD distribution). The use of the AD-CD system PARMLIB and PROCLIB is not required, of course, and you should adjust these names for your needs.

1. Create key storage data sets:

- Edit SYS1.SAMPLIB(CSFCKDS).
  - Complete the JOB statement.
  - Find the VOLUME(XXXXXX) parameter and change it to VOLUME (ZBSYS1).<sup>2</sup>
  - Submit the job for execution and check the results.
- Edit SYS1.SAMPLIB(CSFPKDS).
  - Make the same changes and run the job.
- Edit SYS1.SAMPLIB(CSFTKDS). This data set is optional.<sup>3</sup>
  - Make the same changes and run the job.

2. Copy SYS1.SAMPLIB(CSFPRM00) to ADCD.Z111.PARMLIB(CSFPRM00).

- Edit this member to verify that it is as follows:

```
CKDSN(CSF.CSFCKDS)
PKDSN(CSF.CSFPKDS)
COMPAT(NO)
DOMAIN(0)                <-- You may need to add this line
SSM(NO)
KEYAUTH(NO)
CHECKAUTH(NO)
TRACEENTRY(1000)
USERPARM(USERPARM)
REASONCODES(ICSF)
```

3. Copy SYS1.SAMPLIB(CSF) to ADCD.Z111.PROCLIB(CSF).

- Edit it as follows:

```
//CSF  PROC
//CSF  EXEC  PGM=CSFMAIN,REGION=8M,TIME=1440
//CSFLIST DD  SYSOUT=A,LRECL=132,BLKSIZE=132,HOLD=YES
//CSFPARM DD  DSN=ADCD.Z111.PARMLIB(CSFPRM00),DISP=SHR
```

4. Edit the ADCD PARMLIB member IKJTSO00 and add the following:

```
AUTHPGM NAMES(          /* AUTHORIZED PROGRAM NAMES      */ +
                                     ...
CSFDAUTH          /*THIS WAS ALREADY IN OUR AD-CD SYSTEM */ +
CSFDPKDS         /*WE ADDED THIS ONE           */ +
                                     ...
AUTHTSF NAMES     /*PROGRAMS.....            */ +
                                     ...
CSFDAUTH          /*THIS WAS ALREADY IN OUR AD-CD SYSTEM */ +
```

<sup>2</sup> ZBSYS1 is the volume of the AD-CD z/OS 1.11 system that contains system-related VSAM data sets. The use of this particular volume is not required. If you plan to carry over your cryptographic configuration to future releases of z/OS, then you should place these data sets on a local volume.

<sup>3</sup> It is not used in our basic setup and operation instructions, but may be used for more advanced cryptographic functions. These VSAM data sets are small; we included this one for completeness.

```
CSFDPKDS      /*WE ADDED THIS ONE      */ +
```

Be certain to copy the plus signs at the end of each line!

5. On the MVS console enter the command S CSF. You should see CSF start. It will have a number of error messages but should eventually say ICSF INITIALIZATION COMPLETE.

Later, when you stop z/OS, you may need to add a P CSF command to your shutdown script (or enter it manually).

6. Go to ISPF option 6 and enter the command @ICSF. This should produce the first ICSF panel, similar to that shown in Figure 13-1.

```
HCR7751 ----- Integrated Cryptographic Service Facility-----
OPTION ==>
Enter the number of the desired option.

  1 COPROCESSOR MGMT - Management of Cryptographic Coprocessors
  2 MASTER KEY MGMT - Master key set or change, CKDS/PKDS Processing
  3 OPSTAT           - Installation options
  4 ADMINCNTL       - Administrative Control Functions
  5 UTILITY          - ICSF Utilities
  6 PPINIT          - Pass Phrase Master Key/CKDS Initialization
  7 TKE              - TKE Master and Operational Key processing
  8 KGUP            - Key Generator Utility processes
  9 UDX MGMT        - Management of User Defined Extensions

Licensed Materials - Property of IBM
5694-A01 Copyright IBM Corp. 1989, 2008. All rights reserved.
US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Press ENTER to go to the selected option.
Press END   to exit to the previous menu.
```

Figure 13-1 First ICSF panel

On this panel, select option 1. This should produce a display similar to Figure 13-2. This panel verifies that the coprocessor is active.

```
----- ICSF Coprocessor Management ----- Row 1 to 1 of 1
COMMAND ==>                               SCROLL ==> PAGE

Select the coprocessors to be processed and press ENTER.
Action characters are: A, D, E, K, R and S. See the help panel for details.

  COPROCESSOR      SERIAL NUMBER      STATUS
  -----          -
.  E00              42-00740          ACTIVE
***** Bottom of data *****
```

Figure 13-2 Verify that the cryptographic coprocessor is online

Use F3 to return to the first ICSF panel and select option 6. This option allows you to enter a *pass phrase* that is then automatically used to initialize the basic coprocessor master keys. (An alternative method for initializing master keys is to use multiple other functions on the

ICSF panels. Unless you are familiar with cryptographic coprocessor management, we suggest that you use the simple pass phrase initialization process we describe here.) Complete the pass phrase panel as shown in Figure 13-3 (using your own pass phrase, of course). You need to enter your pass phrase, two data set names, and a character to select the *Initialize system* option. After executing the panel function, you should see the PKA SERVICES ARE ENABLED message in the upper right corner.

```

----- ICSF - Pass Phrase MK/CKDS/PKDS Init          PKA SERVICES ARE ENABLED
COMMAND ==>>

Enter your pass phrase (16 to 64 characters)
====> Bill's secret pass phrase_____

Select one of the initialization actions then press ENTER to process.

X Initialize system - Load the AES, DES and asymmetric master keys to all
  coprocessors and initialize the CKDS and PKDS.

  CKDS ==>> 'CSF.CSFCKDS'
  PKDS ==>> 'CSF.CSFPKDS'

_ Reinitialize system - Load the AES, DES and asymmetric master keys to
  all coprocessors and make the specified CKDS and PKDS the current key data
  sets.
  CKDS ==>>
  PKDS ==>>

_ Add coprocessors - Initialize additional online coprocessors with the
  same AES, DES and asymmetric master keys.

_ Add AES MK - Add the AES master key to all active coprocessors and the
  current CKDS.

Press ENTER to process.
Press END   to exit to the previous menu.

```

Figure 13-3 Pass phrase panel

When we exited from this panel we received an 047 ABEND. (This was in the AD-CD z/OS 1.10S and 1.11 systems.) However, the CKDS function appeared to have been completed correctly. APAR OA30025 may address this problem, but we did not apply it. We assume this problem will be resolved in future z/OS releases.

Return to the first ICSF panel and select option 1 again. Overtyping the initial period in front of the E00 coprocessor name with the letter S and pressing Enter should produce a display similar to that shown in Figure 13-4 on page 146.

```

----- ICSF - Coprocessor Hardware Status -----
COMMAND ==>                                SCROLL ==>
                                           CRYPTO DOMAIN: 0

REGISTER STATUS                            COPROCESSOR E00

Crypto Serial Number                       : 42-00740
Status                                     : ACTIVE
DES Master Key
  New Master Key register                   : EMPTY
  Verification pattern                      :
  Hash pattern                             :
                                           :
  Old Master Key register                   : EMPTY
  Verification pattern                      :
  Hash pattern                             :
                                           :
  Current Master Key register               : VALID
  Verification pattern                      : 2D7BDF2F5A7ADF9C
  Hash pattern                             : 071B3CB4761DCDE4
                                           : E1D3675B90E977C7
AES Master Key
  New Master Key register                   : EMPTY
  Verification pattern                      :
  Old Master Key register                   : EMPTY
  Verification pattern                      :
  Current Master Key register               : VALID
  Verification pattern                      : BCB28DF1FA0FC8EF
Asymmetric-Keys Master Key
  New Master Key register                   : EMPTY
  Hash pattern                             :
                                           :
  Old Master Key register                   : EMPTY
  Hash pattern                             :
                                           :
  Current Master Key register               : VALID
  Hash pattern                             : 9197A99AD60F87AF
                                           : 1E2FD4EB0F59B932

Press ENTER to refresh the hardware status display.
Press END   to exit to the previous menu.

```

Figure 13-4 Coprocessor status

This completes the basic cryptographic coprocessor setup. We suggest you do not experiment with option 2 (MASTER KEY MGMT) functions unless you know what you are doing.

## 13.4 Operational notes

You must start CSF (on the MVS console) in order to use cryptographic functions through ICSF; this is normally the MVS command S CSF. When stopping z/OS you must issue a corresponding stop command such as P CSF. These commands could be added to the various VTAMAPPL startup/shutdown scripts in the z/OS AD-CD system.

You can easily verify CSF operation by going to `omvs` and issuing the following command:

```
od -An -N4 -td /dev/random
```

If CSF (or the emulated cryptographic coprocessor) is not working, the result is an internal error message. If these functions are working, a random number is displayed.

The 1090 cryptographic coprocessor emulation functions are intended for use by developers who require these functions. It should be clearly understood that these emulated functions are not intended to produce a secure system or to function as a secure peer when dealing with private data.

The 1090 stores the coprocessor internal data in the `~/z1090/srdis` directory. There is a subdirectory here for each defined coprocessor; the master keys and other functional data are stored in the subdirectory. The data formats in these records are not documented, but they should not be considered cryptographically secure.

If you used a pass phrase for initialization, you should record the exact characters used (including upper or lower case, spaces, and punctuation). You would need this to recreate the same master keys if you reinitialize the cryptographic functions or want to create duplicate keys on another System z.

### 13.4.1 Multiple zPDT instances

zPDT may have multiple instances in operation. These instances may have shared facilities, such as shared DASD. If shared facilities are used, then a zPDT controller instance<sup>4</sup> must be present as described in “Multiple zPDT instances” on page 71. Coprocessors defined in the controller instance may be shared by all zPDT instances. A maximum of 16 coprocessors may be present in a “normal” zPDT instance. A maximum of 64 coprocessors may be defined for a controller instance.

An additional devmap statement:

```
domain <member name> a y #(a is a coprocessor number, y is a domain number)
```

is used in the devmap of a 1090 instance that is using coprocessors defined in the controller instance. The member name is required if the domain statement appears in the controller instance; it is used if the domain statement is in an operational instance devmap. The domain number (y in the statement above) can be a single number, a list of numbers separated by commas, or a range of numbers separated by a dash. The angle brackets around the member name are not part of the syntax; they indicate an optional parameter here. Remember that the domain numbers must be specified in the ICSF startup parameters and will be different for each z/OS instance/

Three shared cryptographic coprocessors, used by three 1090 instances, might be defined as follows:

```
#----- controller instance -----  
[system]                #no processor is defined for the controller  
...  
...  
[adjunct-processors]  
crypto 0  
crypto 1  
crypto 2
```

<sup>4</sup> Very briefly, a controller instance is a zPDT instance (with a separate devmap and started with an `awsstart` command) that does not contain any System z processors.

```

#----- 1090 instance 1 -----
[system]
processors 1
...
[adjunct-processors]
domain 0 1
domain 1 1
domain 2 1

#----- 1090 instance 2 -----
[system]
processors 1
...
[adjunct-processors]
domain 0 2                #All the domain statements
domain 1 2                #could be in the controller
domain 2 2                #devmap instead. Your choice.

#----- 1090 instance 3 -----
[system]
processors 1
...
[adjunct-processors]
domain 0 3                #If the domain statements are in the
domain 1 3                #controller devmap, they need the relevant
domain 2 3                #member name as the first parameter.

```

Notice that each 1090 instance has a different domain number specified in the domain statements. In this example the domain numbers are the same as the instance numbers, but this is just a coincidence.

## 13.4.2 Coprocessor control commands

A number of commands are included for specialized management of the cryptographic coprocessors. These commands are issued from a Linux terminal window. zPDT must be operational for these commands to be used. They are not needed for normal system usage and we suggest you do not experiment with them unless you have a fairly good understanding of what you are doing. In the following commands the *n* variable is the cryptographic coprocessor number and the *y* variable is a domain number. Briefly, the commands are:

```

$ ap_zeroize -a n -d y
$ ap_zeroize -a n -i

```

This command reinitializes (zeros) all the data, such as keys, that is retained by the coprocessor. In the syntax shown here, *n* is a crypto adapter number and *y* is a domain name. The first version of the command affects only the specified domain. The second version (with the **-i** operand) zeros the whole adapter. Either **-i** or **-d y** must be specified (with an appropriate domain number for *y*).

```

$ ap_query
$ ap_query -a n

```

This command queries basic status and domain information. With no operand it lists the coprocessors available to the System z. With an operand, it lists which domains are used by the indicated coprocessor.

```
$ ap_create -a n
```

This command creates a new (emulated) cryptographic coprocessor.

```
$ ap_destroy -a n
```

This command removes the indicated coprocessor process if it is not connected to a CP process.

```
$ ap_von -a n
```

```
$ ap_von -a n -d y
```

```
$ ap_voff -a n
```

```
$ ap_voff -a n -d y
```

These commands vary online or vary offline connections between coprocessors and their processing queues. The optional *y* operand specifies a domain number.

```
$ ap_vpd -a n
```

This command lists vital product data for the indicated coprocessor.

When a 1090 instance is started (while processing the devmap) an **ap\_create** is issued for that instance. If this is a stand-alone 1090 instance, **ap\_von** commands are issued for all domains. (It is not issued for a controller instance.) If this is a 1090 instance using shared coprocessor resources, **ap\_von** commands are issued for the coprocessors and domains specified in the devmap.

The “real” cryptographic coprocessors on large System z machines have similar control functions, but they are performed in different ways. Do not attempt to use these commands, as listed here, on larger machines.

### 13.4.3 New z/OS releases

The coprocessor master keys, stored in the Linux `srd` subdirectory, must be consistent with the data in the CSF.CSFCKDS and CSF.CSFPKDS data sets in z/OS. If you install a new z/OS release and create new z/OS data sets (while keeping older master keys for the coprocessor functions) then CSF initialization will fail.

For long-term cryptographic usage, you should place the CSF.CSFCKDS and CSF.CSFPKDS data sets on local volumes that will be used with all the releases of z/OS that you might want to invoke.

If you have mismatched master keys and z/OS data sets, you need to zeroize the appropriate coprocessor and domains and then enter a new pass phrase to start over. This, of course, invalidates any existing lower-level keys. If you plan to work with encrypted data (as opposed to simply developing programs that use encryption functions) you need to carefully plan backups for the coprocessor data (in the `srd` subdirectory) and the z/OS data sets used by CSF. The 1090 functions have no special way to recover lost encryption keys.

### 13.4.4 Programming with ICSF

Following is a trivial program that uses the cryptographic coprocessor (via an ICSF programming interface) to obtain random numbers:

```
//OGDENYZ JOB 1,OGDEN,MSGCLASS=X
```

```

//A EXEC ASMACLG,PARM.C='NOXREF',PARM.L='NOLIST,NOMAP'
//C.SYSIN DD *
        PRINT      NOGEN
ICSF AA  CSECT
ICSF AA  AMODE     31
ICSF AA  RMODE     24
        STM      14,12,12(13)      SAVE CALLER'S REGISTERS
        LR       12,15             USE ENTRY-POINT BASE REGISTER
        USING   ICSF AA,12
        LR       2,13             GET A(CALLER'S SAVEAREA)
        LA      13,SAVEAREA       GET A(MY SAVEAREA)
        USING   SAVEAREA,13      MORE 'USING' SPACE
        ST      2,SAVEAREA+4     CHAIN OLD TO NEW
        ST      13,8(2)         CHAIN NEW TO OLD
*
* OPEN FILES AND CHECK RESULTS
*
A1      OPEN      (PRINTD,(OUTPUT))
        TM      PRINTD+48,X'10'   CHECK SYSPRINT OPEN STATUS
        BZ      ERROR1
*
* GET RANDOM NUMBERS AND PRINT THEM
*
        LA      7,20             GET 20 RANDOM NUMBERS
LOOP1   CALL      CSNBRNG,(RETC,REASC,EXDL,EXD,FORM,RANNUM)
        CLC     RETC(4),SZEROS
        BNE     ERROR2
        LA      1,RANNUM         WHERE TO START HEX CONVERSION
        BAL     10,AHEXLINE
        MVC     PRINTLNE(80),SBLANKS
        MVC     PRINTLNE(21),=C'RANDOM NUMBER (HEX) ='
        MVC     PRINTLNE+22(16),SWOUT
        PUT     PRINTD,PRINTLNE
        BCT     7,LOOP1
CLOSEALL CLOSE (PRINTD)
RETURN  L      13,4(13)         GET A(CALLER'S SAVE AREA)
        LM      14,12,12(13)     RESTORE CALLER'S REGISTERS
        SR      15,15           SET RETURN CODE
        BR      14             EXIT
*
* SIMPLE ERROR HANDLING.
*
ERROR1  WTO     'UNABLE TO OPEN SYSPRINT DD STATEMENT'
        B      RETURN
*
ERROR2  WTO     'NON-ZERO RETURN CODE'
        B      RETURN
*
PRINTD  DCB     DSORG=PS,MACRF=(PM),DDNAME=SYSPRINT,LRECL=80,          X
        RECFM=FB,BLKSIZE=8000
* VARIOUS WORK AREAS AND CONSTANTS
PRINTLNE DC CL80' '
RETC     DC     F'0'           RETURN CODE (ICSF)
REASC    DC     F'0'           REASON CODE (ICSF)
EXDL     DC     F'0'           EXIT DATA LENGTH (ICSF)

```

```

EXD      DC      CL4' '          EXIT DATA (ICSF)
FORM     DC      CL8'RANDOM '    RULE FORM
RANNUM   DC      2F'0'          RANDON NUMBER
*
          DROP 12
SAVEAREA DC      18F'0'
SW1      DC      D'0'          WORK AREAS FOR UTILITY ROUTINES
SW2      DC      D'0'          WORK AREA
SPILL    DC      D'0'          SPILL FROM SW2 UNPK INSTRUCTION
SWOUT    DC      CL80' '        OUTPUT AREA
SBLANKS  DC      CL80' '        SOURCE OF BLANKS
SZEROS   DC      2F'0'          SOURCE OF ZEROS
ASCNDECS DC      8F'0'          LOCAL REGISTER SAVE AREA
          SPACE
*-----*
* FORMAT 32 BYTES OF STORAGE INTO HEX.
* INPUT: R1 CONTAINS ADDRESS OF DATA. OUTPUT: 72 BYTES IN SWOUT
*-----*
AHEXLINE STM 1,6,ASCNDECS      SAVE CALLER'S REGS
          LA 2,8                8 WORDS OUTPUT
          LA 3,SWOUT            A(OUTPUT)
          MVC SWOUT(80),SBLANKS
AHEXLINF BAL 5,AHEXLINZ        CONVERT 4 BYTES
          LA 3,9(3)             OUTPUT POINTER
          LA 1,4(1)             INPUT POINTER
          BCT 2,AHEXLINF        LOOP
          LM 1,6,ASCNDECS
          BR 10                 RETURN TO CALLER
*
AHEXLINZ MVC SW1(4),0(1)
          MVI SW1+4,X'00'
          UNPK SW2(9),SW1(5)
          TR SW2(8),AHEXTR-240
          MVC 0(8,3),SW2
          BR 5
AHEXTR   DC      C'0123456789ABCDEF'
          SPACE
          LTORG
          END
/*
//L.SYSLIB DD DISP=SHR,DSN=SYS1.MACLIB
//          DD DISP=SHR,DSN=CSF.SCSFMODE
//G.SYSPRINT DD SYSOUT=*

```

### 13.4.5 z/VM usage

Cryptographic coprocessors are defined for z/VM guests as shown in the following example:

```

USER USERJOE 999999 512M 16E BDEG
  ACCOUNT ABC123 ABC123
  CRYPTO DOMAIN 13 14 15          (domains to be used)
  CRYPTO APDED 1 3                (coprocessors to be dedicated for use)
  OPTION TODENABLE MAINTCCW
  MACHINE ESA 64

```

CPU 0 BASE  
CPU 1  
IPL 190 PARM AUTOCR  
etc

# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

## IBM Redbooks

For information about ordering these publications, see “How to get Redbooks” on page 153. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *IBM System z Personal Development Tool Volume 2 Installation and Use*, SG24-7722
- ▶ *IBM System z Personal Development Tool Volume 1 Introduction and Reference*, SG24-7721
- ▶ *IBM System z Personal Development Tool Volume 4 Coupling and Parallel Sysplex*, SG24-7859
- ▶ *Introduction to the New Mainframe: z/VM Basics*, SG24-7316

## Other publications

- ▶ *System z Personal Development Tool User's Guide and Reference*, G229-1101

## How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks, at this Web site:

[ibm.com/redbooks](http://ibm.com/redbooks)

## Help from IBM

IBM Support and downloads

[ibm.com/support](http://ibm.com/support)

IBM Global Services

[ibm.com/services](http://ibm.com/services)



# Index

## Symbols

/etc/vsftpd.conf 10  
\$SPRT1 command 38

## Numerics

1090 messages 6  
1403 printer 36  
3390 volume, moving 26  
540PAG 53  
540RES 53  
540SPL 53  
540W01 53  
540W02 53

## A

acc command 58  
ADRDSSU job 28  
ADRDSSU restore program 30  
ADRDSSU, utility program 16  
alcckd command, diagnosis 95  
APAR OA27453 43  
AUTOLOG1 functions 59  
aws2scsi command 17  
awscmd device manager 81  
awslog command 96  
awsmount command 37  
awsmount, SCSI tapes 14  
awsprt usage 37  
awsscscsi device manager 14  
awsstart command 3  
awsstart, problems 92  
AWSTAPE file 30  
awstape utilities 17

## B

BLP processing 83  
bonding LAN 114  
BPXTCAFF program 131  
browse command 59

## C

card2tape 17  
card2tape command 17  
CKD versioning 5  
ckdPrint command 95  
client, migration 123  
command  
  acc 58  
  alcckd 95  
  aws2scsi 17  
  awslog 96  
  awsstart 3

browse 59  
card2tape 17  
ckdPrint 95  
cpaccess 59  
cpsyntax 59  
dial 64  
directxa 59  
discard 58  
diskmap 59  
doOSAcmd 96  
dreg 96  
dshrmem 96  
filelist 58  
find\_io 115  
format 58  
gzip 31  
i cms 58  
ind 59  
link 58  
peek 58  
printlog 96  
printrace 96  
purge system prt all 58  
purge system rdr all 58  
q accessed 58  
q all 59  
q alloc all 58  
q alloc map 58  
q da all 58  
q disk 58  
q links 120 58  
q n 59  
q pf 58  
q prt 58  
q stor 59  
q system 58  
rassummary 93  
receive 30  
rel 58  
scsi2tape 17  
senderrdata 92  
tape2file 17  
tapeCheck 96  
tapecheck 17  
top 97  
vmlink 58  
vmstat 96  
x (xedit) 58  
xmit 28  
compressing PARMLIB 43  
core image files 95  
core images, startup 95  
cpaccess command 59  
CPs, CPUs, threads, tokens 2  
CPs, maximum number 2

cpsyntax command 59

## D

DASDVOL, RACF class 127  
DEDICATE statement 64  
def\_reserved\_size, SCSI 16  
DEMOpkg, installation 44  
det command 58  
devmap, for Linux for System z 105  
devmap, z/VM 54  
devmapvm devmap 54  
DHCP 118  
DHCP and local router 118  
DHCP client 118  
dial command 64  
directxa command 59  
disabled waits, z/OS 46  
disc (disconnect) function 56  
discard command 58  
diskmap command 59, 64  
doOSAcmd command 96  
dreg command 96  
dshrmem command 96  
dual boot, example 7  
dumps, z/OS, disk space 20

## E

emulated I/O devices, maximum 9  
Emulex Corporation Zephyr-X LightPulse Fibre Channel  
Host Adapter 16  
ERV=500, in IEAOPTxx 44  
Etherjet PC Card 7  
Ethernet adapters, shared 119  
ethernet, alias usage 114

## F

Fibre SCSI adapter cards 16  
filelist command 58  
find\_io command 6, 115  
force userid 58  
format command 58  
fstab options 131  
ftp activation 10

## G

grub, booting 8  
guest z/OS 60  
gzip command 31

## H

HCD, usage 34  
hckd2ckd command 129–130  
Health Checker 42  
hfa2fba command 129  
htape2tape command 129  
HZSALLCP job 42  
HZSPROC procedure 42

## I

i cms command 58  
I/O devices, number 9  
ICKDSF 30  
IEAOPTxx member 43  
ind command 59  
IODF changes 34  
IPL z.VM 54  
ipl, DEMOpkg 46  
iptables, disable 111  
IRARMCPU, abends 43  
ISPF, starting 42

## J

Java and WAS startup 21  
JES2 print 39  
JES2 setup 38

## K

kernel.msgmni 10

## L

Lexmark printers 36  
link command 58  
link list, addition 43  
Linux monitoring 96  
Local Routers 118  
logs/traces 92  
logstreams, deleting 44  
lpr 38

## M

MAINT userid 57  
MDISK statement 64  
memory, used for I/O devices 9  
message numbers, 1090 6  
messages, 1090 6  
migration, DASD volume 123  
minidisk space 106  
modprobe 16  
mount\_dvd command 52  
msgmni, value 9  
MVS console, lost 41

## N

NICDEF definitions 107  
non-QDIO 114

## O

OAT, multiple instances 72  
OMVS, enabling usage 39  
OPERATOR userid 56  
oprmsg usage 41  
OSA notes 6  
oversize parameter, x3270 20

## P

- paging, with z/VM 65
- Partition Magic 7
- partitioning 8
- PATH parameter, for OSA 6
- PC printer 36
- peek command 58
- PowerQuest Partition Magic 7
- printing 36
- printlog command 96
- printrace command 96
- PRODPK 53
- PROFILE, z/OS TCP/IP 116
- purge system prt all command 58
- purge system rdr all command 58

## Q

- q accessed command 58
- q all command 59
- q alloc all command 58
- q alloc map command 58
- q da all command 58
- q disk command 58
- q links 120 command 58
- q n command 59
- q pf command 58
- q prt command 58
- q rdr 58
- q stor command 59
- q system command 58
- QDIO or non-QDIO operation 7
- QLogic Corp. ISP2432-based 4Gb Fibre Channel to PCI Express HBA 16

## R

- rassummary command 93
- rdrlist 58
- receive command 30
- Red Hat Enterprise Linux for System z 104
- Redbooks Web site 153
  - Contact us x
- rel command 58
- remote operation 9
- RHEL\_5.3 s390x DVD 104
- RMF-III, starting 43
- routers 118
- RRS, disk space 20

## S

- S071 ABEND 20
- SCSI adapter cards, Fibre 16
- SCSI def\_reserved\_size 16
- SCSI tape drives 14, 16
- scsi2tape command 17
- selection menu, 3270 79
- senderrdata command 92, 96
- server, migration 123
- set pf12 retrieve 58

- shell script, printing 37
- SLES-10 SP1 (for System z) 131
- snapdump 92
- SPECIAL statement 64
- spin loop timeouts 20
- SPINLOOPS 4
- Stand Alone Loader 55
- stand-alone dump 24
- SVC dumps, space 21
- SYS1.LOGREC full 40
- SYS1.SAMPLIB for Health Checker 42
- SYSTEM CONFIG, z/VM 106

## T

- tape drives, SCSI 16
- tape2file 17
- tape2file command 17
- tape2scsi 17
- tape2tape 17
- tapeCheck 17
- tapeCheck command 96
- tapecheck command 17
- tapePrint 17
- TCP/IP port 3990 125
- TCPMAINT, for z/VM 108
- telnet activation 10
- threads, Linux 2
- TKE system 141
- token, dates 2
- tokens, multiple 2
- top command 97
- tunnel environment, for OSA 120
- tunnel OSA devices, multiple 114

## U

- ulimit -m and -v 10
- ulimit, memory 9
- USB hardware keys, multiple 3

## V

- VMAC support 75
- vmlink command 58
- vmstat command 96
- vncpasswd, usage 112
- vncviewer, use 111
- Volume migration 123
- VSE420GA.AWS file 67
- vsftp 104
- VSI parameter 4
- VSWITCH function 104
- vswitch support 75

## W

- Websphere Application Server 22
- WebSphere Application Server startup 21
- Windows partition 8
- Windows XP 7
- wsadmin tool 22

## **X**

x (xedit) command 58  
x3270 cursor position 9  
x3270 oversize parameter 20  
xmit command 28  
XOsview program 97  
Xquickstart option 22

## **Z**

z/OS CP and memory display 20  
z/OS operator console, lost 41  
z/VM 5.4 system 51  
z/VM 6.1 installation 51  
z/VM directory 106  
z/VSE installation 67  
zAAP 2  
zFS, in AD-CD system 43  
zFS, stopping 43  
zIIP 2  
ZOSSERV.XMIT file 126  
ZPDTMSRV module 126



**IBM System z Personal Development Tool: Volume 3 Additional Topics**

(0.2"spine)  
0.17"->0.473"  
90->249 pages







# IBM System z Personal Development Tool Volume 3 Additional Topics

## System z Development Tool

## Full z/OS usage

## Linux base

This IBM Redbooks publication introduces the IBM System z Personal Development Tool (zPDT), which runs on an underlying Linux system based on an Intel processor. zPDT provides a System z system on a PC capable of running current System z operating systems, including emulation of selected System z I/O devices and control units. It is intended as a development, demonstration, and learning platform; it is not designed as a production system.

This book, discussing more advanced topics, is the last of three volumes. The first volume introduces zPDT and provides reference material for zPDT commands and device managers. The second volume describes the installation of zPDT (including the underlying Linux, and a particular z/OS distribution) and basic usage patterns. The third volume discusses more advanced topics that may not interest all zPDT users.

The systems discussed in these volumes are complex, with elements of Linux (for the underlying PC machine), z/Architecture (for the core zPDT elements), System z I/O functions (for emulated I/O devices), and z/OS (providing the System z application interface), and possibly with other System z operating systems. We assume the reader is familiar with the general concepts and terminology of System z hardware and software elements and with basic PC Linux characteristics.

## INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

## BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:  
[ibm.com/redbooks](http://ibm.com/redbooks)

SG24-7723-02

ISBN 0738434914