Hints and Tips for z/VSE 6.2

XX00-0000-00

Hints and Tips for z/VSE 6.2

XX00-0000-00

Note !

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS without any warranty or liability. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environment do so at their own risk.

Fourteenth Edition (May, 2018)

This book applies to Version 6 Release 2 of IBM z/Virtual Storage Extended (z/VSE) and earlier releases.

You may send your comments via the Internet or by FAX:

Email to: s390id@de.ibm.com
From the z/VSE homepage: https://www.ibm.com/systems/z/os/zvse
FAX (Germany): 07031-16-3456
FAX (other countries): (+49)+7031-16-3456

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

About This Book

In this publication the z/VSE Development Team has updated existing information where needed and has added new information to reflect the latest z/VSE enhancements.

If this book helps z/VSE customers to better understand new and existing functions and makes it easier for them to analyze and solve problems, it serves its purpose well.

Summary of changes with z/VSE V6.2

The following summarizes the changes with z/VSE V6.2

The product z/VSE V6.2 (5686-VS6) is the new version of the z/VSE operating system.

The z/VSE SIR command displays *z/VSE 6.2.0* and also the RSL PSP bucket for this release is published specifying *VSERSL620*. The upgrade value for preventive service planning for this release is *zVSE620*.

Starting with z/VSE V6.1, z/VSE is delivered as English version only.

CICS TS for z/VSE V2.1 cannot be used with z/VSE V6.2. It is replaced by CICS TS for z/VSE V2.2

Installation

- z/VSE V6.2 can be installed via initial installation
- Fast Service Upgrade to z/VSE V6.2 is possible from z/VSE 6.1.
- Fast Service Upgrade to z/VSE V6.2 from a release prior to z/VSE 6.1 is not possible,
- IBM IPv6/VSE is automatically installed during initial installation.

Updated z/VSE BASE PROGRAMs

- 5686-VS6 z/VSE V6.2
- 5655-VSE CICS Transaction Server for z/VSE V2.2
- 5686-BS1 IBM IPv6/VSE V1.3 - new release is now on the base tape (formerly extended base tape).
- 5686-CS1 IBM TCP/IP for z/VSE V2.2 - new release

Summary of Changes in this Document

This edition of the manual includes minor updates, corrections, and new information as listed below.

The following chapters have been changed to reflect the new z/VSE version.

- **Chapter 1, “System Control Program” on page 1** .
  Information about stand-alone dump program on SCSI disk has been added.
- **Chapter 2, “Internal Attention Routine Commands” on page 47**

  The section **“SIR” on page 49** has been updated.
  The SIR HELP command**“SIR HELP” on page 49** and
  “SIR SMF” on page 59 now show the new operand ZHPF.
- **Chapter 4, “Console” on page 95**

  The section **“CORCMD” on page 97** has been updated. It now shows the CORCMD TRACE.
- **Chapter 5, “Job Control” on page 109**

  The new Job Control program return codes have been described.
- **Chapter 7, “Librarian” on page 161**

  The section **“Restore IJSYSRS - IJSYSRS corrupted” on page 166** is new.
- **Chapter 8, “VSE/VSAM” on page 181**
- **Chapter 9, “VSE/ICCF” on page 213**
  The following section has been added
  - **“View Library Members” on page 216**

In **Chapter 11, “Tape Library Support” on page 227** the following changes took place

- Return and reason codes were removed.
- The *z/VSE 6.2 System Macros Reference* is listed as reference.

Chapter 12, “Capacity Measurement Tool” on page 229 is new.It gives a short introduction to the Capacity Measurement Tool (CMT) with useful references.

In **Chapter 13, “Language Environment for VSE (LE/VSE)” on page 231** several sections have been updated to show the changes with modification level LE/VSE 1.4.10:

- “LE/VSE 1.4.10 in z/VSE 6.2” on page 234
- “LE/VSE Attention Routine Interface and Commands” on page 235
- “LE/VSE Run-time Options” on page 236

Chapter 14, “z/VSE Security and Security Migration” on page 243 has been reworked.

Chapter 15, “CICS TS for z/VSE 2.2” on page 247 has been updated.
The sections

- **“Version dependent functions” on page 247** and
- **“Error Message DFHPA1107 During CICS TS Startup” on page 247** are new.

In **Chapter 18, “Interactive Interface, System Files and Configuration” on page 277** the following sections have been updated:

- “Hints and Tips for Fast Service Upgrade (FSU)” on page 290
  Information about the media for FSU has been added.

The following sections are new:

- “Activate TCP/IP messages in the Online Message File” on page 279
- “CICS TS CSD Migration” on page 284
- “DFHCSDUP Utility” on page 307
- “Apply PTF from the internet” on page 309
- “Transferring Tape Image Files to virtual Tape” on page 310
- “How to define and use auxiliary trace file B” on page 314

Chapter 22, “Debug Tool for VSE/ESA (LE)” on page 331 has been updated, especially the following sections

- “CICS Considerations” on page 332.
- “VSE Partition Requirements” on page 333.

The following chapter has been changed to reflect the new z/VSE version.

- **Chapter 21, “VSE Health Checker” on page 329**

In **Chapter 22, “Debug Tool for VSE/ESA (LE)” on page 331** the section **“CICS Considerations” on page 332** has been updated and shows the correct usage of the DFHCSDUP utility.

In **Chapter 23, “Miscellaneous Hints and Tips” on page 339** the section **“EREP Hints and Tips” on page 341** has been added.

In **Chapter 25, “z/VSE Initial Installation using an Installation Disk” on page 363** the following section is improved:
“Pitfalls” on page 365.

z/VSE BASE - Program Publications

All z/VSE manuals are available as softcopy only.

- IBM Publications Center From the IBM Publications Center, you can download most z/VSE online publications free-of-charge in PDF format.

  `http://www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss`

- z/VSE home page (documentation)

  `http://www.ibm.com/systems/z/os/zvse/documentation/`

- z/VSE Knowledge Center

  `http://www.ibm.com/support/knowledgecenter/SSB27H/zvse_welcome.html`

- CICS TS Knowledge Center

  `http://www.ibm.com/support/knowledgecenter/SSECAB`

Chapter 1. System Control Program

The different components of the z/VSE system carefully check any submitted job to see whether the prerequisites for the execution of the job are fulfilled. The user gets an information message if the Job Control statements are incorrect or incomplete. The Job Control statements assign and allocate system resources to the application program. A message may occur if these resources are not available in the system or if they are assigned to another user.

If an error occurs during the execution of the job, the z/VSE terminator routines provide error messages and cancel the failing program. If the proper dump options are active, the termination routines create a storage dump of the failing partition. This storage dump shows the PSW and the registers at the time of the failure, and it provides a snapshot of the storage contents of the failing partition. In most cases the diagnostic messages, the message explanation and the storage dump are sufficient to find the cause of the malfunction.

Besides the dumps which are automatically invoked by the z/VSE termination routines, there are the attention routine dumps. Via the DUMP command you may request a dump of parts of virtual storage.

If a dump does not give enough information to solve the problem, you may use the interactive trace program to follow selected instructions or all instructions of the program step by step. The interactive trace program may also be used if an application program contains logical errors and you do not understand why the program does not produce the wanted results.

Supervisor or Hardware Malfunctions

If a program malfunction occurs in the z/VSE supervisor itself, then we may or may not get a reasonable error message. Whether the system is able to communicate the problem to the console operator depends on the degree of the malfunction. Whenever the z/VSE system operator is suspecting a system failure, which could be a loop or a system hang condition, then it is his responsibility to intervene and determine the degree of the malfunction. He does so by first trying to issue some useful commands.

- DEBUG ON
- STATUS command, or
- CANCEL AR command
- REPLID

First try to issue the DEBUG ON command to make sure that the z/VSE system will start its internal monitoring capabilities. If that command is not being accepted or responded to, then the attention task is in some abnormal state and you should try the **CANCEL AR** or the **RC** command to get it freed again. In case that doesn't work either, then try the **REPLID** command to check if the console facility is responding. If the REPLID command is being responded to, then proceed as described under “Attention Not Responding” on page 4. If the REPLID command is not being responded to, then

The processor may be in

- a continuous disabled loop, or
- a disabled wait state.

Now it's time to intervene via the processor's monitoring facilities. The procedures to invoke and execute the monitoring facilities are different for different types of processors. Therefore they are not described in this report. For detailed information you may consult the Operations Guide of your Central Processing Complex. In a z/VSE system running under control of z/VM, the VM/Control Program provides the monitoring facilities. See “Important VM/CP Functions” on page 37 which describes the CP commands which monitor the system status of a virtual machine. The first thing you need to do is:

Stop the processor and display the PSW

Determine Hard or Soft Wait

The processor's monitoring facilities offer a function to display the Program Status Word (PSW). In a disastrous error state, the most important bit in the PSW is bit 14 the **Wait** bit. (This is the bit X'02' in the second byte of the PSW). If the **wait bit is zero** then the processor is active and you should conclude that the system is in a disabled loop.

The second full word in the PSW (the PSW address field) shows the instruction address where the processor has stopped. If you start and stop the processor repeatedly, you may be able to determine the scope of the loop. You could also try to activate the processors instruction trace facility to examine some loop details (see also Processor Loop).

If the **wait bit is one**, then the processor is in the wait state. That is, the processor does not execute any instructions. In this wait state the second full word of the PSW does not contain an instruction address but may contain useful information identifying the reason of why that wait state had been entered. Before you can draw such a conclusion, you need to examine bit 6 and 7 of the first byte of the PSW. If either of these two bits X'02' (I/O mask bit) or X'01' (external mask bit) is ON, then the processor is in a **SOFT WAIT** (see Soft Wait States). The processor is in a **HARD WAIT** if both these bits are OFF (see also Hard Wait States) Figure 1 shows the bit positions of the above mentioned PSW bits:

02xxxx00 xxxxxxxx	I/O mask bit: If this bit is one, then the processor is enabled for I/O interruptions
01xxxx00 xxxxxxxx	External mask bit: If this bit is one, then the processor is enabled for external interruptions
xxx2xx00 xxxxxxxx	Wait bit: If the wait bit is one, then the processor is in the wait state

Figure 1. Important PSW bits

Processor Loops

Processor loops can have many causes. See some examples below and what you should do in case you are experiencing such a processor loop.

The Processor Runs in a Continuous Loop

Symptoms: The processor executes instructions but the z/VSE system seems to be idle. It does not respond to attention commands. When you stop and start the processor several times, the PSW is always active (the wait bit in the PSW is zero) and the PSW address field shows different addresses at any stop. The processor does not perform useful work; it is in a continuous loop.

What To Do: Collect information about the loop, and take a stand-alone dump

This is probably a situation where a stand-alone dump followed by a new IPL is the only measure to recover the z/VSE system. But before you take the stand-alone dump, retrieve some data about the processor loop and do a STORE STATUS afterwards. It is useful to find out whether the loop is short or long. Stop the processor, enter the single instruction mode, and press the start key several times. The processor executes single instructions and displays the address of any executed instruction. If the loop is short, then note the instruction addresses within the loop. If the loop is long, then note selected addresses within the loop. For that, you reset the processor status to normal instruction processing and stop and start the processor several times. Probably you will find that the processor loops between two or three blocks of processor storage. Any addresses occurring within the loop will later be useful to interpret the stand-alone dump data.

Processor Interrupt Loop

Symptoms: The processor loops on the same instruction.

What To Do: Take a stand-alone dump.

The worst case of a processor loop is a one-instruction interrupt loop. Such a loop may occur if a program with PSW key zero overwrites the low storage area. For example, a program that writes location X'1D0' with binary zeros. That will inevitably lead to a disaster error. On the next programming interruption (e.g. on a page fault), the processor stores the actual PSW into location x'150' and fetches a new PSW from location X'1D0'. Since location X'1D0' does not contain a valid PSW, the processor recognizes another programming interruption. The processor enters a loop, continuously fetching the same incorrect PSW from location X'1D0'. If the z/VSE system runs under control of z/VM, then the Control Program VM/CP interrupts the virtual machine and displays the message "CP ENTERED; PROGRAM INTERRUPT LOOP". The only possible operator action for a processor interrupt loop is a stand-alone dump followed by a new IPL of the z/VSE system.

Device Check Loop '07E6'

Symptoms: Check storage location 0. If you see **X'07E6cuu'** then you have detected a device check loop as explained here.

What To Do: Remove one ADD statement and re-IPL.

A disabled instruction loop may occur during IPL, if an added device does not respond to SSCH requests. The IPL program issues a SENSE command to any device for which an ADD statement exists. The IPL program moves the value

X'07E60cuu' to low storage location zero and checks the device status via a STSCH and a TSCH command. If the device does not answer, then the processor may run into an endless loop. These events show you that the IPL program is in a 'device check loop':

- the IPL program showed message 0J10I RESTART POINT BYPASSED as last message on the screen.
- the low storage location zero shows the value X'07E60cuu'
- the value 0cuu in byte 2 and 3 does not change (IPL always waits for the same device).

There is no way out of the disabled loop. The only way to bring up the system is to repeat the IPL procedure without the critical ADD statement.

SDAID Monopolizes the z/VSE System and Cannot Be Stopped

Symptoms: You started SDAID and can't do anything while SDAID is running.

What To Do: Reset control register 9 to zero.

This SDAID problem looks like a processor loop, but the system is not in a loop. You specified a comprehensive instruction or branch trace, e.g. an instruction trace with AREA=ALL and ADDRESS=0:*. The z/VSE system is 100% busy; it executes instructions and processes programming interruptions all the time. The z/VSE system will not even accept a STOPSD or ENDSD command. In this situation use the processor's monitoring facilities to reset the tracing conditions in the first byte of control register 9. (The bits X'80', X'40', and X'20' control the branch trace, the instruction trace, and the storage alteration trace, respectively). If you have reset the relevant bits to zero, the z/VSE system will return to reasonable work and accept a STOPSD or an ENDSD command.

Attention Not Responding

You have issued the REPLID command and it did provide you the appropriate response, but the z/VSE system did NOT RESPOND to, or even accept z/VSE AR-commands (0D14I COMMAND IGNORED) and the **CANCEL AR** or the **RC** command did not free the Attention task either and you have made sure that Attention is not waiting on an I/O request which may require operator intervention, or the CANCEL cuu(,FORCE) command, then its about time to take a stand-alone (SA) dump followed by a new IPL to recover your z/VSE system. Please make sure that you have done a STORE STATUS prior to taking the SA Dump.

System Status

You have entered the DEBUG ON command and the Attention routine has replied with 1I40I READY. Now do the following.

First check the status of the system via the Attention command STATUS. A STATUS command (without any operand) shows which tasks are ready to run and which tasks are in a bound state. Examine the waiting tasks and check the events they are waiting for. You may find out that a set of tasks (or even all tasks) are bound in a deadlock. You break the deadlock situation by canceling the partition which holds the critical resources.

In case the STATUS command doesn't indicate any failure symptoms, you should issue the DEBUG SHOW(,ALL) command (see DEBUG) and examine its output to eventually find a failure.

Concurrent Microcode Upgrade

For information about concurrent microcode upgrade in z/VSE please refer to z/VSE home

`http://www.ibm.com/systems/z/os/zvse/about/status.html#microcode`

Soft Wait States

Soft Wait State (Waiting for I/O Completion)

Symptoms: If the processor is in the wait state (PSW bit 14 is one) and the processor is enabled for I/O interruptions (PSW bit 6 is one), then the system is in a soft wait state.

What To Do: Check system status carefully before taking a stand-alone dump.

The processor is waiting for an I/O interruption. In this situation check the activity of the system carefully before you start any recovery actions. The fact that you see the same soft wait PSW on any processor stop does not necessarily mean that the z/VSE system is idle. The z/VSE system may do reasonable work, e.g. copy some hundred cylinders from one disk to another disk. If you stop the processor during such an action several times, you may always see the same soft wait PSW. Only when you are sure that the soft wait is solid should you start recovery actions.

You can activate the processor by pressing the reset key (to remove pending device conditions) and entering an attention command, e.g. a STATUS command. The processor will process the interruption. If the processor falls back into the soft wait state and you don't see any actions after a while, then you should take the same actions which are proposed for a disastrous hard wait situation. Take a stand-alone dump and re-IPL the system. In case of an I/O wait, the low core locations 0 to 7 do not contain values important for problem determination. An exception from that general rule may occur during IPL.

Soft Wait '08C1' During IPL

Symptoms: Check storage location 0. If you see X'08C10cuu' then you have detected the softwait as explained here.

What To Do: Ready the referenced device.

If an 'INTERVENTION REQUIRED' condition occurs during IPL and the I/O message writer is not yet available, then the IPL routine enters the value X'08C10cuu' into low core location zero and enters a soft wait state. The only action required at that time is to make the referenced device ready. The IPL routine will continue processing. These events show you that the IPL program is in an 'intervention required' wait:

- the IPL program showed message 0J10I RESTART POINT BYPASSED as last message on the screen.
- the low storage location zero shows the value **X'08C10cuu'**
- the value 0cuu in byte 2 and 3 does not change (IPL always waits for the same device).
- 0J62I ACTUAL CHANQ IS *number*.

Soft Wait (Waiting for an External Interruption)

Symptoms: If the processor is in the wait state (PSW bit 14 is one) and the processor is enabled for external interruptions (PSW bit 7 is one), then the system is in this soft wait state.

What To Do: Check the PSW and perform the action the system is waiting for.

The processor is waiting for an external interruption. In this situation the contents of the PSW and the storage locations 0 to 4 define the required action. Soft waits with a PSW enabled for external interruptions are often programmed waits. The system expects an action from the operator, followed by an external interruption. On some processors you generate an external interruption via the IRPT key; on other processors you enter the monitoring facilities and select the option INTERRUPTION. On a VM controlled system you issue the CP command EXTERNAL.

- **PSW = 01060000 00000000 00000000 0000EEEE**

  The SDAID program has created this soft wait PSW. A TRACE statement has been specified with the HALT option. The specified event has occurred and SDAID has entered a soft wait state. The operator may display some storage locations using the processor's monitoring facilities. The operator has two choices to resume normal operation

  – generate an external processor interruption, e.g. CP EXTERNAL when running under z/VM. SDAID continues tracing and stops on the next traced event (again with a soft wait PSW of 01060000 00000000 00000000 0000EEEE)

  – set the value x'FF' into low core location 0 and generate an external processor interruption.
  SDAID will reset the HALT option and continue tracing.

- **PSW = 01060000 00000000 0000000 00EEEEEE**

  The SDAID program has created this soft wait PSW. The SDAID output device (OUTDEV) needs attention. The low core bytes 2 to 3 contain the device address (cuu) of the SDAID output device. The low core locations 0 to 1 display one of three information codes:

  62C1 End of tape reel

  62C5 Intervention required. Printer or tape is not ready

  62E2 Intervention required. Byte 4 contains an error recovery action code

  The operator should handle the device problem (mount new tape, feed paper to printer, etc.) and then generate a processor interruption.

  If it is not possible to lift the 'intervention required condition', then generate a processor interruption, too. If you generate an external interruption and the SDAID output device is still unready, SDAID switches tracing off and the z/VSE system resumes operation without tracing. Then stop SDAID explicitly via a STOPSD command. SDAID will display the message 4C01A SDAID ALREADY STOPPED. This message describes the I/O problem and gives you the possibility to dump the SDAID area. Then issue the ENDSD command to terminate SDAID finally.

Hard Wait States

Symptoms: If the PSW has one of the states as shown below, then the system has entered a hard wait condition.

What To Do: Check the PSW and re-IPL the system with or without a stand-alone dump.

If the PSW wait bit (bit 14) is one and the mask bits (bits 6 and 7) in the PSW are zero, then the processor is in a hard wait state. A hard wait state is not necessarily caused by a software or hardware malfunction. Often the hard wait state signals the normal completion of a stand-alone process. If the stand-alone dump program or the stand-alone restore program terminate processing, they enter a hard wait state. This is a normal end of processing, and you should not think of taking a stand-alone dump. The proposed action is to IPL the z/VSE system. The following lines show the different hard wait codes and the proposed operator actions.

- **PSW = 00020000 00000000 00000000 00001000**, or
- **PSW = 00020000 00000000 00000000 00EEEEEE**

  These hard wait PSWs show that a disaster error has occurred in the VSE/AF supervisor. The error may have occurred during IPL or during normal operation. If the address part of the hard wait PSW contains 00EEEEEE then the RAS routines have generated the hard wait state.

  In any case it is important to check the low core locations 0 to 7. These bytes give information about the cause of the error. The chapter 'VSE/Advanced Functions Wait Codes' of the z/VSE message manual describes the meaning of the error codes found at low core locations 0 to 7. In a hard wait state with one of the above shown PSWs, the system is dead and there is no hope for recovery. The only proper action is to display the PSW, display low storage locations 0 to 7, take a stand-alone dump, and IPL the z/VSE system.

- **PSW = 00020000 00000000 00000000 00000FFF**

  This hard wait PSW indicates that z/VSE has successfully processed a *signal quiesce* event. A signal quiesce event is presented by the hardware when a disruptive operation is to be performed or when SIGNAL SHUTDOWN is issued for the guest under z/VM. If z/VSE is enabled for this kind of event the hardware or z/VM will grant z/VSE a grace period to shutdown before finally executing the disruptive operation. z/VSE will issue message **0W01D** to allow the operator or a console automation to initiate a controlled system shutdown as a response to message 0W01D.

- **PSW = 00020000 00000000 00000000 00CExxxx**

  The stand-alone dump program has created this hard wait PSW. When the stand-alone dump program completes processing, it enters a hard wait state with the above PSW. The value xxxx in the address portion of the PSW shows the result of stand-alone dump processing. An address value of 00CE0000 signals successful completion. Other address values, like 00CE0100 (I/O error during tape IPL) or 00CE0080 (end of extent on stand-alone dump disk file), signal that the stand-alone dump program terminated abnormally.

  These error codes from the rightmost two bytes of the hard wait PSW are explained in the chapter 'VSE/Advanced Functions Wait Codes' of the message manual. The recommended action after any stand-alone dump hard wait is to re-IPL the z/VSE system.

- **PSW = 00020000 00000000 00000000 0000EEEE**

  The DEBUG program has created this hard wait PSW. The Address Compare Stop event of the DEBUG feature has generated the above hard wait code. The operator has defined a DEBUG address compare stop, and the contents of the specified field matches the user specified pattern. **This is not an error Situation**. The operator established an address stop and the requested address stop came along.

  You may analyze the system status and then restart the z/VSE system using the RESTART feature provided by the processor's monitoring facilities. The DEBUG feature has prepared a restart PSW in low core location 0. If the z/VSE system runs under control of z/VM, then use the CP command 'SYSTEM RESTART'. The DEBUG feature resets the address stop. The z/VSE system resumes operation as if no stop were encountered.

- **PSW = 04060000 00000000 00000000 00002000**

  The stand-alone supervisor has created this hard wait PSW. When a program running in the stand-alone environment completes and the operator decides to terminate the stand-alone environment, then the z/VSE system enters the above hard wait PSW. **This is not an error situation**. The stand-alone environment came to its normal end. There is no reason to take a stand-alone dump. The recommended action at this time is to re-IPL the z/VSE system.

- **Other hard wait codes**

  The z/VSE system does not produce hard wait codes other than those mentioned above. If a z/VSE system enters a hard wait state and the address portion of the PSW contains another value, then the hard wait state was probably caused by a vendor product. In this case you should check the product documentation of the vendor products installed in your z/VSE system. In order to document the hard wait problem, note the PSW, take a stand-alone dump and re-IPL the z/VSE system.

Diagnostic Messages

The z/VSE system detects a problem and issued an error message

- read message explanation, and
- perform the recommended actions

The previous section discussed severe system errors, characterized by the fact that the processor looped or entered a wait state. This section discusses error situations where the z/VSE system keeps running and displays documented error messages.

The z/VSE message manual "Messages and Codes" interprets the messages issued by the z/VSE package and its component licensed programs. The manual explains the reason for the message, it describes the action the system has taken, and it gives a recommendation for the proper programmer and operator response. The description of any message may be retrieved on-line via the On-line Message Explanation function (OME). Simply move the cursor to the message number and press function key 9.

The z/VSE error messages contain a message number and the message text. Often the message text contains variables, like a return or reason code, an SVC number, a phase name, or a function code. These enclosed variables provide

important diagnostic data. If you bring a z/VSE problem to the attention of your IBM service location, always report the message number **and** the complete message text.

If a z/VSE job fails, then all Job Control statements and all system responses may be important for the analysis of the problem. You get all JCL messages logged on the console if you include a // LOG statement immediately after the // JOB statement. This // LOG statement directs all JCL messages to the console and into the hard copy file. All logged messages are accessible via the REDISPLAY command, and they can be printed at a later time via the PRINTLOG utility program. If you pass an error report to an IBM service location, always submit the complete console log with all Job Control statements and all system messages to the service location.

Message 0P31A DEVICE NOT OPERATIONAL

What To Do: Try to wake up device via 'ONLINE cuu(,FORCE)'.

The above message occurs if an I/O unit is broken, power is switched off, or a device with the referenced address does not exist. You may get message 0P31A if you attach a device to your processor after IPL has completed. The IPL routines issue I/O commands to all possible unit addresses. If a device does not respond to the I/O command, then the IPL program may flag it internally as 'not operational'. Such late attachments often occur if a z/VSE system runs under control of z/VM or in an LPAR environment, or if I/O devices are shared between systems via control unit or channel path switching.

If you attach an I/O device after IPL has completed, z/VSE will automatically recognize that the device(s) became operational and will thus initiate its ONLINE processing. You may want to issue the 'ONLINE cuu(,NOASN)' to tell the z/VSE system explicitly that the device is present and ready to execute I/O operations. NOASN is an option that will prevent devices (tapes) to be assigned to a particular system (LPAR).

Message 0J28I LOCK FILE ON cuu: NUMBER OF SHARING CPUS EXCEEDED

What To Do: Format the lock file by temporarily specifying the parameter TYPE=F on the IPL DLF command and optionally increase the NCPU parameter value.

The above message occurs if the lock file, used to share DASDs between a group of two or more z/VSE systems, does not have a free system entry in its header to accommodate the subject z/VSE system. The lock file header has a fixed amount (2 to 31) of system entries, that is defined by the NCPU paramter of the IPL DLF command during the formatting of the lock file. Each z/VSE system that is part of the DASD sharing group occupies one system entry in the lock file header.

There are several possible reasons for running out of system entries in the lock file header, for instance:

- Adding a new z/VSE system to the DASD sharing group requires an additional system entry. If the initial NCPU parameter value specified during formatting of the lock file did not account for this new system, the lock file needs to be re-formatted while increasing the NCPU parameter value.
- Migrating a z/VSE system to a new machine (which comes along with a new CPU ID) causes two system entries to be used. One system entry remains

reserved for the z/VSE system having used the old CPU ID and another one is reserved for the system using the new CPU ID. In this case there are two options: either clear the old system entry (i.e. CPU ID) from the lock file header using the UNLOCK command or format the lock file to clear all entries.

To clear a single system entry from the lock file refer to the description of the UNLOCK command in the z/VSE System Control Statements manual. Special precaution must be taken to ensure that the system being cleared from the lock file is not operating under any circumstances. In LPAR the HMC/SE Deactivate or Reset Clear task should be performed on the LPAR. Under z/VM the guest should be logged off.

To format the lock file perform the following steps:

1. Shutdown all z/VSE systems of the DASD sharing group using the lock file. Take care that the systems are not operating under any circumstances while performing the following steps. In LPAR the HMC/SE Deactivate or Reset Clear task should be performed on the LPARs. Under z/VM the guests should be logged off.
2. IPL one single z/VSE system with load parameter '..P' to request prompting for IPL parameters. The load parameter may be specified in LPAR in the HMC/SE Load panel or under z/VM using the LOADPARM operand of the IPL command.
3. Reply 'STOP=DLF' to message
   0I03D ENTER SUPERVISOR PARAMETERS OR ASI PARAMETERS
   to request prompting new DLF command. The DLF command from the IPL procedure will be displayed on the console as reference. It will be ignored if a new DLF command is specified.
4. Transcribe the DLF command displayed on the console while changing the TYPE parameter to TYPE=F (or specifying TYPE=F if the TYPE parameter has been omitted), to request formatting of the lock file, and optionally increase the NCPU parameter value to account for the maximum number of z/VSE systems in the DASD sharing group.
5. Reply 'DELETE' to the message
   0I80D DOS.LOCK.FILE ON *cuu:* DUPLICATE NAME ON VOLUME *volid* or
   0I37D DOS.LOCK.FILE ON *cuu:* OVERLAP ON UNEXPIRED FILE DOS.LOCK.FILE.
6. If the lock file has been successfully formatted, IPL the other z/VSE systems of the DASD sharing group.

Refer to the description of the DLF command in the z/VSE System Control Statements manual for further details.

Error When Opening a File

What To Do: Ensure that the file descriptions match.

When the z/VSE I/O routines open a file, they take file descriptions from three different sources. An error occurs if these different descriptions do not match.

1. The first source of information is the file description (called the 'DTF') in the application program. The DTF describes the file name, the file type and it provides some file characteristics. You can check the DTF only if you have a source listing of the failing application program.

2. The second source of information is the system's label area. This label area contains the DLBL statement and the EXTENT statement(s). You include label statements into your job stream or use label statements which have been put permanently into the system's label area in a previous job. The label statements describe the file name, the expiration date, and the track allocations. If the label statements are not contained in the failing job stream, use the LSERV utility program to retrieve the contents of the system's label area.
3. The third source of information is the VTOC, the Volume Table of Contents. The VTOC entry contains the file name, and it shows the tracks which are occupied by the file. The VTOC entry depends on the DLBL and EXTENT statements which where active when the file was created. You use the LVTOC program to list the contents of a disk pack. The LVTOC program needs assignments for SYS004 and SYS005.

You compare the output of the LSERV program with the output of the LVTOC program (and eventually with the DTF found in the application program). If you detect a mismatch in the different descriptions of the file, then correct the labels and rerun the job.

```
// JOB   LVTOC
// ASSGN SYS004,cuu          address of disk unit
// ASSGN SYS005,cuu          address of printer unit
// EXEC LVTOC                list Volume Table of Contents
/&
```

Figure 2. LVTOC sample job

Phase Not Found

What To Do: Search the active libraries.

You specified a program name on an EXEC, an LFCB or an LUCB statement, or issued a LOAD macro within an application program. An information message or a return code informs you that the referenced phase name is not contained in the allocated program libraries. Use the Job Control LIBLIST statement to retrieve the actual library chain, and then use the Librarian program to inspect the contents of the assigned libraries. The SEARCH function checks whether a single phase is contained in a library, and the LISTDIR function prints the names of all library members.

```
// JOB   LIBRSRCH
// EXEC LIBR
SEARCH IJBSDUMP.PHASE  L=IJSYSRS     search a phase in a library
LISTDIR  L=IJSYSRS                   list all member names
/&
```

Figure 3. Search a library sample job

Corrupted Library

What To Do: TEST the library and repair or restore it (if necessary).

The structure of a system library or a private library may get damaged, if a disaster error occurs in the librarian program LIBR or in an implicitly called librarian routine. The librarian program LIBR provides a TEST feature to check the status of a library.

If the TEST function shows errors, then rerun the TEST function with the REPAIR option. If another TEST run still shows errors, then back up the library (if you do not already have a backup tape), and restore the library. If a following TEST run still shows errors, then contact the IBM support center and have the results of the TEST run available.

See also “Test of Libraries” on page 166 for more information.

```
// JOB   LIBRTEST
// EXEC LIBR
TEST L=IJSYSRS [REPAIR=YES]          test library integrity
/&
```

Figure 4. Test a library sample job

Message 4936I NO MORE AVAIL/MATCH XTNT

What To Do: Check if one of the following applies

- History File is too small, or
- History File is corrupted

The MSHP program issues the message 4936I. Any RETRACE listing shows a percentage of how much assigned space is used. If the percentage is higher than 80% then the history file is probably too small. It is recommended that you extend the history file. CREATE a larger history file and copy the old file into the new one.

If you get the message 4936I and the percentage is lower than 80 %, then the history file is probably corrupted. You have to take actions to recreate a correct history file.

- Use a backup of the history file if available. Re-application service or re-installation of a product may be necessary to reflect the latest changes to the system.
- If no backup of the history file is available, perform a MERGE of the system history file into a newly created history file. All history file information is transferred from the old history file to the new one, and the chain of free records is re-built. In many cases, this solves the problem.
- If no backup is available, and the MERGE could not solve the problem then the history file needs to be repaired. The history file analysis can be done by Level 2 support. Using an output of the "DUMP HISTORY SYSTEM" function and the 'MSHP Diagnosis Reference Manual (LY33-9153)', the record chains must be followed, until the invalid pointer is found. Once found, the DITTO program can be used to repair the record chain.

Message 0S06I A DUMP MACRO WAS ISSUED (Info/Analysis)

What To Do: Do the following

- increase partition size to at least 2 megabytes

  Most problems occurring in Info/Analysis are caused by a shortage of GETVIS storage. If the partition is too small, you may get the above mentioned error message 0S06I or another BLN message explaining that an internal or external function is failing.

  The GETVIS requirement of Info/Analysis depends on the size of the processed dump, on the size of the files BLNDMF and BLNXRTN, and on the storage requirements of the invoked exit routines, like DFHDAP. It is recommended that you run Info/Analysis in a partition with at least two megabytes of storage. It does, however, not help if you increase the SIZE parameter on the EXEC statement from SIZE=300k to a higher value. A SIZE=400k, for example, does just the opposite. SIZE=400k leaves 100k of bytes of the partition storage unused and decreases the available GETVIS storage by 100k.

- decrease size of Info/Analysis files BLNDMF and BLNXRTN

  In the standard z/VSE systems, the dump management file BLNDMF and the external routines file BLNXRTN are defined with rich contingency. Even very big z/VSE installations do not require such big files. It seems that three tracks of a 3380 disk are sufficient for the dump management file BLNDMF, and that one track of a 3380 disk is sufficient for the external routines file BLNXRTN. Sometimes Info/Analysis reads these files - in total - into the partition GETVIS storage. You improve the space performance and the time performance of Info/Analysis if you decrease the size of BLNDMF and of BLNXRTN.

- check names of loaded dumps

  Ensure that the dump names you define for loaded dumps do not conflict with system generated dump names. Problems may occur from the following two sources:

  – You loaded two dumps whose name differ only in the first character or you loaded a stand-alone dump into a sub library and named it, let's say, JACK0001. When Info/Analysis processes this dump, it generates a mapping member and calls it MACK0001. (That's the way Info/Analysis builds the names of auxiliary library members. Info/Analysis generates a mapping member and an extension member and generates names for them by replacing just the first character). If you on-load another dump and name it PACK0001 then you will no longer have unique library member names. Info/Analysis searches the dump library for a mapping member named MACK0001 which may have been created in a previous Info/Analysis session. Info/Analysis will erroneously use the same mapping member MACK0001 for all dumps with names like BACK0001, JACK0001, PACK0001, and so on. This will lead to severe errors with different misleading error messages.
  – You loaded a dump whose name is similar to a system generated dump name or you loaded a stand-alone dump and named it PE300007. Names of this form may conflict with dump names generated by the z/VSE termination routines. z/VSE creates dump names of the form DBG00001, DX700023, SF300004, SE300007, where the bytes 2 and 3 denote the partition-id and bytes 4 to 8 are a decimal number between zero and 99999. If you generate a dump with a name like PE300007, then

Info/Analysis will create a mapping member for this dump which may conflict with mapping members created for ABEND dumps.

Solution: You avoid all naming problems if you introduce the following naming convention:

- do not use D, H, L, M, N, S, X as the first character of a dump name. Instead use **one** heading letter, for example the letter 'A' as first letter for all dump names within a dump sub library.
- place at least one alphabetic character in positions 4 to 8 of the dump name.

Message BLN2013I DUMP MANAGEMENT FILE IN ERROR

What To Do: Scratch the file, and run Info/Analysis UTILITY function.

The Info/Analysis Dump Management File BLNDMF is in error or it has been overwritten erroneously by another program. Before you build a new dump management file, you have to purge the defective file. Use the DITTO function PVT to scratch the VTOC entry of the file BLNDMF. (The DITTO function PVT runs in interactive mode only.)

If the DITTO program is not available in your installation, use another method to get rid of the old file BLNDMF (e.g. overwrite the file with a file of a different name and answer 'DELETE' on the overlap file message).

Then specify labels for the dump management file, and call the Info/Analysis function UTILITY to create and format a new dump management file. On the next invocation of the dump management function, Info/Analysis scans the directory of the dump library SYSDUMP and builds entries for all available dumps.

Problem Management Programs and Tools

This chapter deals with the programs and functions available for error detection and error recording. It describes the following dump and trace features:

- the ABEND dump
- the DUMP command
- the Stand-Alone Dump program
- the Interactive Trace program
- the SDAID program
- the DEBUG feature
- the VM/CP TRACE function

ABEND Dump

If an application program terminates abnormally, then the z/VSE termination routines write an error message on the console and pass control to the dump routines. The z/VSE dump routines check the Job Control options to decide whether a dump is to be generated and to which output medium the dump should be directed. The operator or the programmer define JCL options which control the amount of dump data and the dump destination.

The JCL options DUMP, PARTDUMP, and NODUMP control the amount of dump data.

- option DUMP dumps the supervisor, important parts of the shared space, and the user partition.
- option PARTDUMP generates a dump of the user partition.
- option NODUMP suppresses the dump generation.
- options SYSDUMP and NOSYSDUMP control the dump destination.
- option SYSDUMP directs the dump to the dump library.
- option NOSYSDUMP directs the dump to SYSLST.

The dump output is also printed on SYSLST, if SYSDUMP is specified but the dump library is not defined (LIBDEF statement missing), or the dump library is full. If the dump is written into a dump library it can be processed via Info/Analysis.

DUMP Command

The operator uses the DUMP command to dump selected parts of the z/VSE system to a printer device or to a tape device. It is not possible to enter the dump output into the POWER list queue or to enter the dump directly into a dump library. If the dump is written on a tape device, then the dump tape may be loaded into the dump library.

If an SDAID session is active, you may dump the current contents of the trace buffer to a tape unit. It is recommended that you freeze the SDAID buffer via a STOPSD command, before you dump the buffer to tape. But you should not issue the ENDSD command before you have dumped the buffer contents. The ENDSD command returns all used resources to the z/VSE system and clears the SDAID buffer. The program DOSVSDMP formats the trace entries and prints them on SYSLST. Figure 5 on page 16 shows the usage of the DUMP command.

dump x7,5a7000-5a7fff,00e	dump an address range in the partition X7
dump b000-cfff,00e	dump an address range in the shared space
dump sva,280	dump selected parts of the Shared Virtual Area
dump sup,280	dump the z/VSE supervisor
dump dspace,myspace,bg,0-f00,00e	dump X'F00' bytes of the data space "MYSPACE" allocated to BG
dump buffer,480	dump SDAID buffer to tape
dump F7,0-7fffffff,480	dump the partition F7 together with the shared space (supervisor & SVA)

Figure 5. Attention DUMP command to Printer or Tape

Stand-alone Dump Program

If your z/VSE system enters a disastrous hard wait state or runs into a continuous processor loop, then no normal system action is possible. In order to collect information about the cause of the error you invoke the stand-alone dump program to save the status of the failing z/VSE system.

The following steps are required to produce a stand-alone dump:

1. Create stand-alone dump program on disk or tape.
2. Set appropriate stand-alone dump options.
3. Take the stand-alone dump by IPLing the stand-alone dump program on disk or tape.
4. Transfer stand-alone dump data from disk or tape to IBM for problem analysis.

Create Stand-alone Dump Program on Disk or Tape

You have the choice of performing a stand-alone dump to disk or to tape. The recommended storage medium for the stand-alone dump is a disk device other than DOSRES and SYSWK1.

The stand-alone dump program on disk is created with the DOSVSDMP utility in one of the following ways:

1. Submit a job similar to the following.

```
// JOB SADMPEBC  CREATE STANDALONE DUMP PROGRAM ON ECKD DISK EBC
* ********************************************************************
* * DLBL and EXTENT statements for dump file IJSYSDU
* ********************************************************************
// DLBL IJSYSDU,'VSE.DUMP.FILE'
// EXTENT ,,,,68550,75000   !!    modify     !!
* ********************************************************************
* * DLBL and EXTENT statements for dump program file IJSYSDI
* ********************************************************************
// DLBL IJSYSDI,'VSE.DUMP.PROGRAM'
// EXTENT ,,,,1,7           !! do NOT modify !!
// EXEC DOSVSDMP,PARM='CREATE DUMP DEVICE=EBC' !!    modify     !!
/*
/&
```

Figure 6. Sample Job to Create a Stand-alone Dump Disk - ECKD

```
// JOB SADMP204  CREATE STANDALONE DUMP PROGRAM ON FBA DISK 204
* ********************************************************************
* * DLBL and EXTENT statements for dump file IJSYSDU
* ********************************************************************
// DLBL IJSYSDU,'VSE.DUMP.FILE'
// EXTENT ,,,,276,20000000  !!    modify     !!
* ********************************************************************
* * DLBL and EXTENT statements for dump program file IJSYSDI
* ********************************************************************
// DLBL IJSYSDI,'VSE.DUMP.PROGRAM'
// EXTENT ,,,,2,256         !! do NOT modify !!
// EXEC DOSVSDMP,PARM='CREATE DUMP DEVICE=204' !!    modify     !!
/*
/&
```

Figure 7. Sample Job to Create a Stand-alone Dump Disk - FBA

```
// JOB SADMP204  CREATE STANDALONE DUMP PROGRAM ON FBA SCSI DISK
* ********************************************************************
* * DLBL and EXTENT statements for dump file IJSYSDU
* ********************************************************************
// DLBL IJSYSDU,'VSE.DUMP.FILE'
// EXTENT ,,,,276,20000000  !!    modify     !!
* ********************************************************************
* * DLBL and EXTENT statements for dump program file IJSYSDI
* ********************************************************************
// DLBL IJSYSDI,'VSE.DUMP.PROGRAM'
// EXTENT ,,,,2,4095        !! do NOT modify !!
// EXEC DOSVSDMP,PARM='CREATE DUMP DEVICE=5C5' !!    modify     !!
/*
/&
```

Figure 8. Sample Job to Create a Stand-alone Dump Disk - SCSI

IJSYSDI contains the stand-alone dump program, which gets control, when the dump disk is IPLed. The stand-alone dump program writes dump data records (with a fixed length of 4112) to the sequential file IJSYSDU. In Figure 6 the dump file IJSYSDU is defined on ECKD disk EBC with 75000 tracks corresponding to 5000 cylinders. In Figure 7 the dump file IJSYSDU is defined on FBA disk 204 (actually a VM-emulated FBA on SCSI) with 20000000 blocks.

2. Use the Interactive Interface dialog *Create Standalone Dump Program on Disk* (SYSA fast path 462) to create and submit a job similar to the ones in Figure 6 and Figure 7.
3. Execute the DOSVSDMP utility without any additional parameter and follow the prompts.

Notes:

1. During execution DOSVSDMP initializes each dump data record in IJSYSDU, which can take a long time to complete. However this job can run in background and will not affect other workload. Once the job has completed the stand-alone dump disk can be IPLed at any time a stand-alone dump is required.
2. The stand-alone dump program must match the z/VSE version and release. Hence it must be recreated if the z/VSE version or release changes.

Set Appropriate Stand-alone Dump Options

The dump data records written by the stand-alone dump program are organized as logical files inside the (physical) dump file IJSYSDU. When the dump disk is IPLed, they are written to IJSYSDU in the following sequence:

Figure 9. Logical Dump Files Inside IJSYSDU

Number	Data Dumped	Remarks
001	SUPERVISOR+SVA	Mandatory, supervisor and shared virtual area
002	PMRAS-R	Mandatory, real storage areas used by Page Manager
003	PMRAS-nn	Mandatory, one or more files with Page Manager address spaces
mmm	xx-PARTITION	Optional, files with partition address spaces
nnn	xx-DATA_SPACE	Optional, files with partition data spaces
ooo	xx-MEMORY_OBJ	Optional, files with private memory objects
ppp	SHARED-MEMORY_OBJ	Optional, file with shared memory objects

SUPERVISOR+SVA, PMRAS-R and PMRAS-nn are always the *first* files to be written to IJSYSDU. The *last* one to be written is SHARED-MEMORY_OBJ, provided shared memory objects had been defined in the running system and STDOPT SADMPSMO=YES is in effect.

The sequence and number of dump files in between are controlled by // OPTION SADUMP=(m,n,o) as specified in the respective partition.

m Determines whether the partition's address space is to be included in IJSYSDU

n Determines whether the partition's data spaces are to be included in IJSYSDU

o Determines whether the partition's private memory objects are to be included in IJSYSDU

The corresponding standard option is by default STDOPT SADUMP=(0,0,0), which means, that *no* partition-related entities are to be written to IJSYSDU.

Note that the system-provided startup jobs for POWER, CICS, VTAM, Basic Security Manager and TCP/IP already specify `// OPTION SADUMP=5` to have their address spaces included in IJSYSDU. Use `QUERY OPTION,xx` to display the currently active options for partition xx.

The value of m,n,o determine the sequence or order of the partition-related entities on IJSYSDU. First all entities with m or n or o equal to 9 are dumped, then the ones equal to 8 and so on. This may be useful when the disk space allocated to IJSYSDU is not large enough to hold all desired dump files and becomes full. By specifying a high digit for the most important dump files you can increase the probability that they are contained in IJSYSDU before the full-condition arises.

You are better off having enough dump space available. Arithmetically

- one cylinder (or 15 tracks) on IJSYSDU gives 720KB dump space (ECKD)
- 1000 blocks on IJSYSDU gives 444KB dump space (FBA)

If you only want to have the first three mandatory files included in IJSYSDU then a dump space of 300 cylinders or 500000 blocks should be sufficient in most cases. Note that even the first (and for problem determination most important) file `SUPERVISOR+SVA` varies in size and depends for example on the system GETVIS area and its usage.

If you want to provide enough disk space for *all* entities (address and data spaces, private memory objects) then an upper limit for the required dump space is VSIZE as displayed in the TOTAL line of the `MAP VIRTUAL` command output. Taking the VSIZE value may be a waste of disk space, a better estimate is the difference *TOTAL minus AVAIL*, however the AVAIL value changes when dynamic partition or data spaces are (de-)allocated. Unreferenced pages of virtual storage do not claim dump space, so even *TOTAL minus AVAIL* might be too high.

As a rule of thumb take

- 1000 cylinders or 1600000 blocks per GB *TOTAL minus AVAIL*, if you want to provide enough space for *all* entities
- 300 cylinders or 500000 blocks, if you want to provide space for at least `SUPERVISOR+SVA` and `PMRAS-xx` which is the minimum requirement for problem determination.

Take the Stand-alone Dump by IPLing the Stand-alone Dump Program on disk

In case you need to take a standalone dump, store the CPU status and IPL the stand-alone dump disk as follows:

IPL an ECKD or FBA dump disk

Make sure you store the status before IPLing the stand-alone dump disk:

- If running under z/VM, do a *CP STORE STATUS* before you IPL the stand-alone dump disk. See *z/VM CP Commands and Utilities Reference.*
- If running in LPAR mode, on the HMC LOAD panel select
  - the *Store Status* checkbox
  - *Normal* as load type

IPL a SCSI dump disk

First, select the FCP adapter used for IPL of the SCSI disk. Preferable use a previously configured dump-only adapter (FIXME see *Administration manual*). If none is available, use any FCP adapter configured for use by z/VSE. Note that this might result in an incomplete system dump.

1. If running under z/VM
   a. set the SCSI dump disk with CP SET DUMPDEV, ie.

      ```
      CP SET DUMPDEV PORT <wwpn> LUN <lun> BOOTPROG 1
      ```

   b. IPL the FCP adapter

      ```
      <fcp_cuu> DUMP
      ```

2. If running in LPAR mode
   a. on the HMC LOAD panel select

      ```
      load type 'SCSI dump'
      load address = <FCP cuu>
      wordwide portname = <WWPN>
      logical unit number = <LUN>
      Boot program sector = 1
      Boot record block address = 0
      ```

While the stand-alone dump program is active, you will see messages similar to the following:

```
4G34I STAND-ALONE DUMP IN PROGRESS ON DISK 0EBC
4G45I START DUMPING OF PARTITION FB
4G45I START DUMPING OF PARTITION F7
4G45I START DUMPING OF PARTITION F3
4G45I START DUMPING OF PARTITION F2
4G45I START DUMPING OF PARTITION F1
4G10I STAND-ALONE DUMP COMPLETE
```

Figure 10. Sample Output of Stand-alone Dump Program

The stand-alone dump is complete as soon as you see message `4G10I`. You can then IPL the system's DOSRES disk to bring the z/VSE system up again. In Figure 10 the partitions FB, F7, F3, F2 and F1 specified `// OPTION SADUMP=(5,0,0)` to have their address space included in the stand-alone dump.

While the stand-alone dump program is active, your system is down and not available for users. The time needed for the stand-alone dump program to complete depends on

- the number and size of the entities to be dumped: the more and the larger entities to be dumped the more time required until stand-alone dump program completes.
- the ratio of virtual storage to real storage: retrieving pages from the page data set is more time-consuming than retrieving pages from memory.

If stand-alone dump processing takes more time than expected or acceptable, you may decide to interrupt the stand-alone dump processing after the first few dump files have been created (`SUPERVISOR+SVA`, `PMRAS-R`, `PMRAS-00`). To interrupt the stand-alone dump processing, just IPL the system's DOSRES disk. Be aware that the dump data may be incomplete in this case.

Transfer Stand-alone Dump data from disk to IBM for problem analysis.

There are following options:

1. Transfer SYSDUMP library members
   - Use the Interactive Interface dialog *Scan Dump Files on Disk* (SYSA fast path 465) to create and submit a job similar to the following.

```
// JOB SCANDMPD  SCAN DUMP FILE ON DISK
* ********************************************************************
* * DLBL and EXTENT statements for dump file IJSYSDU
* ********************************************************************
// DLBL IJSYSDU,'VSE.DUMP.FILE'
// EXTENT ,,,,68550,75000
// EXEC DOSVSDMP,PARM='SCAN DEVICE=EBC'
/*
/&
```

Figure 11. Sample Job to Scan Logical Files on a Stand-alone Dump Disk

The job output is similar to the following:

```
DUMP FILE  DUMP TYPE  NAME      DATE        DATA DUMPED
---------  ---------  --------  ----------  -------------------------
   001     SADUMP               2012/05/17  SUPERVISOR+SVA
   002     SADUMP               2012/05/17  PMRAS-R
   003     SADUMP               2012/05/17  PMRAS-00
   004     SADUMP     SECSERV   2012/05/17  FB-PARTITION
   005     SADUMP     TCPIP00   2012/05/17  F7-PARTITION
   006     SADUMP     VTAMSTRT  2012/05/17  F3-PARTITION
   007     SADUMP     CICSICCF  2012/05/17  F2-PARTITION
   008     SADUMP     POWSTART  2012/05/17  F1-PARTITION
   009     SADUMP               2012/05/17  SHARED-MEMORY_OBJ
END OF DUMP
```

Figure 12. Logical Files on a Stand-alone Dump Disk

   - Onload logical files from the dump stored on the disk into the dump library (see “Restrictions for Onloading Dump Files” on page 23 below). Use the Interactive Interface dialog *STORAGE DUMP MANAGEMENT* (SYSA fast path 43) to create and submit a job similar to the following.

```
// JOB DMPONL12  ONLOAD DUMP FROM DISK
// ASSGN SYS009,EBC
// DLBL IJSYSDU,'VSE.DUMP.FILE'
// EXTENT SYS009,,,,68550,75000
// EXEC PROC=DTRINFOA
// EXEC INFOANA,SIZE=300K
   SELECT DUMP MANAGEMENT
     DUMP NAME SYSDUMP.BG.SUPSVA
          RETURN
   SELECT DUMP ONLOAD
          VOLID DISK SYS009
          FILE 1 LAST
          RETURN
   SELECT END
/*
/&
```

Figure 13. Sample Job to Onload File 1: SUPERVISOR+SVA

If there is not enough space left in the dump library, you need to enlarge it, see “Extend the VSE Dump Library” on page 300 for details. To extend the dump library, you can also use the skeleton SKDMPEXT which is stored in VSE/ICCF library 59.

- FTP (in binary) the dump library member(s). First you need to define the SYSDUMP library to TCP/IP.

TCP/IP for z/VSE (CSI)

```
DEFINE FILE,PUBLIC='SYSDUMP',DLBL=SYSDUMP,TYPE=LIBRARY
```

IPv6/VSE (BSI)

```
INPUT LIBRARY lib sublib member type mode
for example
INPUT LIBRARY SYSDUMP BG SUPSVA DUMP FIXED
```

In Figure 13 the library member to be transferred is `SYSDUMP.BG.SUPSVA.DUMP`. When sending PMR-related stand-alone dump files to IBM please use binary file transfer (`BIN`) and following naming convention:

`xxxxx.bbb.ccc.yyy.zzz`

e.g. `34143.055.724.sad001.bin`. In this example, 724 is the country code for Germany.

Figure 14. File Naming Conventions

	Explanation	Example
xxxxx	1st part of PMR number	34143
bbb	2nd part of PMR number (branch office)	055
ccc	3rd part of PMR number (country code)	724
yyy	A short description for the file (e.g. file 001 of stand-alone dump disk)	sad001
zzz	File extension	bin

You may further compress the dump using ZIP. In this case use file extension zip.

2. Transfer IJSYSDU file

   First you need to provide label information. In the EXTENT and ASSGN statement you need to specify the same operands as in Figure 6 on page 17 or Figure 7 on page 17.

   ```
   // DLBL IJSYSDU,'VSE.DUMP.FILE'
   // EXTENT SYS020,,1,0,68550,75000
   // ASSGN SYS020,EBC
   ```

   For FBA-SCSI devices you need the following definition

   ```
   // DLBL IJSYSDU,'VSE.DUMP.FILE',,SD,CISIZE=4608
   ```

   Insert these three statements into

   - the TCP/IP startup job, when using CSI FTPD
   - the batch job, when using CSI FTP or FTPBATCH
   - the batch job starting the FTP client or server, when using BSI

   Then you need to define the IJSYSDU file to TCP/IP.

   TCP/IP for z/VSE (CSI)

   ```
   DEFINE FILE,PUBLIC='IJSYSDU',DLBL=IJSYSDU,TYPE=SAM,RECFM=F,LRECL=4112
   ```

   IPv6/VSE (BSI)

   ```
   INPUT SAM IJSYSDU BLKSZ 4112 RECSZ 4112 RECFM F
   ```

   When sending a PMR-related IJSYSDU file to IBM please use binary file transfer (`BIN`) and follow the naming conventions described in Figure 14 on page 22.

Restrictions for Onloading Dump Files

1. INFOANA as executed in Figure 13 on page 22 cannot handle dump files larger than 1GB.
   Circumvention: Use DITTO TL as documented in the *Diagnosis Tools* manual.
2. LIBR cannot handle members larger than 2GB. As a consequence, memory objects larger than 2GB can be dumped, but cannot be loaded to the dump library.
   Circumvention: transfer the whole IJSYSDU file.

Considerations for Stand-alone Dump on Tape

DOSVSDMP creates the stand-alone dump program as the first file on an unlabeled tape. Put the cartridge aside for a possible error situation. If you IPL the tape in order to take a stand-alone dump, the stand-alone dump program writes the dump data to new tape files on the same tape. Already existing tape files from a previous stand-alone dump will be overwritten.

After taking the stand-alone dump, you will find following files on the tape:

Figure 15 (Page 1 of 2). Dump Files on Stand-alone Dump Tape

Number	Data Dumped	Remarks
001	No dump data	Mandatory, stand-alone dump program
002	No dump data	Mandatory, work area used by stand-alone dump program

Figure 15 (Page 2 of 2). Dump Files on Stand-alone Dump Tape

Number	Data Dumped	Remarks
003	SUPERVISOR+SVA	Mandatory, supervisor and shared virtual area
004	PMRAS-R	Mandatory, real storage areas used by Page Manager
005	PMRAS-nn	Mandatory, one or more files with Page Manager address spaces
mmm	xx-PARTITION	Optional, files with partition address spaces
nnn	xx-DATA_SPACE	Optional, files with partition data spaces
ooo	xx-MEMORY_OBJ	Optional, files with private memory objects
ppp	SHARED-MEMORY_OBJ	Optional, file with shared memory objects

"Set Appropriate Stand-alone Dump Options" on page 18 describes, how the `// OPTION SADUMP=(m,n,o)` command controls the sequence and number of the optional dump files.

If the complete stand-alone dump does not fit on one single cartridge, the stand-alone dump program erases the last incomplete file and requests a new cartridge by message

```
4G36I  END OF VOLUME ON DUMP TAPE cuu. MOUNT NEW TAPE OR RE-IPL VSE
```

After the operator mounts a scratch tape the stand-alone dump program continues processing the entity which did not fit on the previous cartridge.

If a dump entity is bigger than the capacity of one single cartridge, the stand-alone dump program will split the dump entity into several files. The symptom record at the end of the cartridge shows `DUMP_CONTINUED_ON_NEXT_TAPE` in its section 5.

Note: Depending on your VSIZE 3480 cartridges (200-400MB) or even 3490E cartridges (800-2400MB) might be no good choice for stand-alone dumps. Use 3590 or 3592 cartridges instead.

Errors Occurring during Stand-alone Dump Program

If an unrecoverable I/O error occurs on the tape device, the stand-alone dump program issues the message

```
4G37I ERROR ON DUMP TAPE cuu. MOUNT NEW TAPE OR RE-IPL VSE
```

and enters an enabled processor loop. As soon as you mount a new cartridge and ready the tape unit, the dump program continues dumping on the next cartridge.

If a disaster error occurs during stand-alone dump processing, the stand-alone dump program enters a hard wait state with a hard wait PSW of the form

```
PSW = 00020000 00000000 00000000 00CExxxx
```

The value xxxx in the address portion of the PSW shows the cause of the error.

The error codes xxxx from the rightmost two bytes of the hard wait PSW are explained in the chapter 'VSE/Advanced Functions Wait Codes' of the z/VSE message manual. The recommended action after any stand-alone dump hard wait is to re-IPL the z/VSE system.

Capacity Problems on Tape

In most cases 3590/3592 cartridges have enough capacity to hold the entire virtual storage (VSIZE) of a z/VSE system.

Nevertheless the tape resident stand-alone dump is not limited to the capacity of one single cartridge. If the cartridge becomes full then you may continue dumping on one or more other cartridges.

When a dump tape becomes full, the stand-alone dump program issues message 4G36I and enters an enabled processor loop. As soon as you mount a new cartridge and ready the tape unit, the stand-alone dump program continues dumping on the next cartridge. If the stand-alone dump program wrote its output to more than one cartridge, then use the SCAN option of DOSVSDMP to find out which files are contained on the different cartridges. The stand-alone dump program ensures that all cartridges contain complete dump files. The dump files do not span cartridges.

Real Dump

The stand-alone dump program normally takes a 'virtual' dump. It uses the region, segment and page tables to dump the different address spaces with ascending virtual addresses. In addition it uses the page data set on disk to retrieve the pages which are not in processor storage at the time of the dump being taken.

The stand-alone dump program can only provide a virtual dump if a minimum of prerequisites is fulfilled. If the System Communications Region, the region, segment and page tables or the DPD table are destroyed, then the stand-alone dump program can only dump the real pages. That is, it dumps the pages which are presently in storage without address translation. Such a real dump may be helpful to analyze supervisor problems (since the supervisor is not pageable and its virtual addresses do not differ from the real addresses), but it is nearly useless to analyze problems in the private spaces.

If a Stand-alone Dump Does Not Help

The stand-alone dump gives a snapshot of the z/VSE system at the time when the operator IPLed the stand-alone dump device. The stand-alone dump shows the system's memory, but it sometimes does not show how the error originated, because it contains only little trace back information.

For example if the machine-check new PSW is overlaid by a faulty program, then this won't hurt unless a machine check occurs. Between the overlay and the incoming machine check hours or even days might have passed by. The stand-alone dump will only reveal the overlaid storage, but not the culprit.

If the information contained in the stand-alone dump is not sufficient to solve the problem and if the problem is likely to occur again, activate a trace program to monitor important system data before the system runs into the disastrous situation. You may choose between the SDAID program and the DEBUG feature. If the error is 'solid' (you know a way to reproduce the problem whenever you execute a specific program or a series of programs), then use the SDAID trace program. If the error occurs intermittent once a day or even once a week, then it is not feasible to run a time-consuming SDAID trace. In this case it is preferable to activate the internal DEBUG trace feature.

In case of rare and intermittent problems always issue the Attention Routine command `DEBUG ON` to provide more detailed trace back data for problem determination.

Interactive Trace Program

The interactive trace program traces the execution of application programs running in static or dynamic user partitions. It operates at the level of machine instructions and virtual storage addresses, similar to the CP debugging facilities in z/VM. The z/VSE users may activate the trace program independently in different partitions. The invoked tracing function is active for the duration of one job step.

You activate the interactive trace program via the EXEC statement. The TRACE parameter on the EXEC statement defines - as default - an instruction trace for all partition storage.

```
// EXEC ABCD,TRACE
```

You control tracing via the interactive trace commands. Figure 16 shows the interactive trace commands. The lower case letters show the possible abbreviation. The TRace command defines or deletes traces. The Query command displays the active traces. The Display and Alter command display and alter storage and registers. The GO command resumes the stopped user program. A reply-id without any parameters is interpreted as a GO command.

TRace Inst ADDRess=41a3f0:42cfff	define instruction trace
TRace BRanch ADDRess=41a3f0.40	define branch trace for an interval of 64 bytes
TRace STor ADDRess=41a3f0	define storage alter trace for a one-byte field
TRace ABend	define abnormal-end trace
TRace END ALL	end all traces
TRace END 3	end the trace with number 3
Query	display all active traces
Display 4035a2	display one 16-byte line of storage data
Display 4035a2.50	display X'50' bytes of storage data
Display 4035a0:4035ff	display a storage interval
Display Gr	display all general purpose registers
Display G12	display general purpose register 12
Display Psw	display the program status word
Alter 4035a2 DATA=fefefe	alter three bytes in storage
Alter g7 DATA=12b	store the value 0000012b into general register 7
go	execute next program instruction(s)
GO OUTPut=SYSLST	switch to batch tracing
GO OPTION=PARTDUMP	modify the dump option

Figure 16. Interactive Trace commands

SDAID

SDAID (System Debugging Aid) is the trace program for z/VSE system routines. It offers a variety of trace types and output options. SDAID collects trace data and writes it on a printer device or on a tape device. The IBM manual z/VSE Diagnosis Tools describes the SDAID program in detail.

Trace Initialization and Execution

An SDAID session consists of two steps: the initialization step (the trace set-up), and the execution step (the tracing). You enter the trace initialization commands via attention routine commands or as SYSIN statements in any free batch partition.

In the attention routine you start the initialization process with the command SDAID. The SDAID command is followed by one OUTDEV command and up to ten TRACE commands. The OUTDEV command defines the SDAID output device. The TRACE command defines the trace type and the tracing conditions. The READY command ends the initialization process.

If you initialize the SDAID session via SYSIN, then you use the statement // EXEC SDAID. This statement is followed by one OUTDEV statement and up to ten TRACE statements. The OUTDEV and TRACE statements entered via SYSIN have the same format as the OUTDEV and TRACE commands. A /* statement completes the initialization process via SYSIN. When the initialization step is complete, the partition is free for the execution of any traced or not-traced program. The trace initialization is easier if you use the SDAID JCL procedures. See some examples of a trace initialization via attention commands, via direct SYSIN statements, and via a JCL procedure below. Note that the three methods in this example initialize exactly the same SDAID session.

The trace execution is controlled via attention commands. The STARTSD (or STRTSD) command starts tracing. The STOPSD command stops the trace data collection temporarily. The ENDSD command terminates the SDAID session finally. It returns all GETVIS space and frees all resources held by the SDAID program.

```
sdaid
BG 0000 4C05I PROCESSING OF SDAID COMMAND SUCCESSFUL
outdev printer=00e
BG 0000 4C05I PROCESSING OF OUTDEV COMMAND SUCCESSFUL
trace inst=* area=f3 address=41207a:4123b3
BG 0000 4C05I PROCESSING OF TRACE COMMAND SUCCESSFUL
ready
BG 0000 4C05I PROCESSING OF READY COMMAND SUCCESSFUL
```

Figure 17. SDAID initialization via ATTENTION command

```
0 // exec sdaid
BG 0000 4C40I SDAID
BG 0000 4C05I PROCESSING OF SDAID COMMAND SUCCESSFUL
BG-0000 4C41I ENTER YOUR SDAID COMMAND
0 outdev printer=00e
BG 0000 4C40I OUTDEV PRINTER=00E
BG 0000 4C05I PROCESSING OF OUTDEV COMMAND SUCCESSFUL
BG-0000 4C41I ENTER YOUR SDAID COMMAND
0 trace inst=* area=f3 address=41207a:4123b3
BG 0000 4C40I TRACE INST=* AREA=F3 ADDRESS=41207a:4123b3
BG 0000 4C05I PROCESSING OF TRACE COMMAND SUCCESSFUL
BG 0000 4C41I ENTER YOUR SDAID COMMAND
0 /*
BG 0000 4C40I READY
BG 0000 4C05I PROCESSING OF READY COMMAND SUCCESSFUL
BG 0000 4C44I ENTER 'STARTSD' ATTENTION COMMAND TO ACTIVATE SDAID
BG 0000 1I00D READY FOR COMMUNICATIONS.
```

Figure 18. SDAID initialization via SYSIN statements

```
0 // exec proc=sdinst,area=f3,address='41207a:4123b3',printer=00e


****** SDAID Execution (always via attention commands) *****
startsd
...
now execute the program you want to trace
...
stopsd
...
startsd
execute another program step
endsd
```

Figure 19. SDAID initialization via JCL procedure

OUTDEV Command

OUTDEV {Printer=cuu|Tape=cuu}

The OUTDEV command defines a printer device or a tape device as SDAID output device. SDAID does not accept a spooled printer device. If the SDAID output is directed to a tape device, then you use the utility program DOSVSDMP to print the contents of the SDAID tape on SYSLST.

OUTDEV Command with Wrap-Around Buffer

OUTDEV [BUffer=nn] [{Printer=cuu|Tape=cuu}]

You use the parameter BUffer only to define a buffer which is filled in wrap-around mode. The value nn specifies the buffer size in units of 1024 bytes. If tape=cuu is specified, the buffer size is restricted to a value between 3 and 32. If the TAPE parameter is not specified, you may specify a value from 3 to 99. SDAID collects trace data in this wrap buffer, but it does not print the data if the buffer becomes

full. It requires explicit actions to print such a wrap-around buffer. Figure 22 on page 35 (example 2-3) shows a method to print the buffer data within an SDAID session. SDAID traces selected instructions in a user partition and moves the trace data into a wrap-around buffer. It prints the buffer contents whenever a program check occurs in this partition.

TRACE Command

The TRACE command defines the trace type, the traced partition or program, the tracing range, the tracing output, and the tracing options. Figure 20 shows several trace statement examples. The numbers in the following description refers to the numbers in the figure.

```
trace branch area=f3 output=(greg dump reg=7:100) option=nojcl           (1-1)
trace branch area=all address=ba398:ba398 output=greg option=(occ=1:5)   (1-2)
trace branch area=bg phase=abcd offset=200:*                             (1-3)

trace cancel area=bg output=(greg dump partition)                         (2)

trace EXT=0040  output=(greg creg lowcore lta pta )                       (3)

trace inst=(d207 92) area=f3 address=41207a:4123b3                       (4-1)
trace inst=* area=f3 address=7312:7bff output=greg option=sup            (4-2)
trace inst=branch jobname=abcd jobnum=134 output=greg                    (4-3)
trace inst=44 area=x9 address=0:*                                        (4-4)

trace io area=bg unit=(360 361) output=(ccwd=500)                        (5-1)
trace io area=all unit=01f output=ccw option=(occurrence=1:200)          (5-2)

trace pgmc=* area=f3 output=(greg ptab ttab dump partition)              (6-1)
trace pgmc=(1 5) jobname=test address=0:* output=(greg lowcore)          (6-2)

trace pgml phase=abcd  output=( greg dump add=400000:420000)             (7)

trace ssch  unit=280 output=tod                                          (8-1)
trace ssch  area=f3 output=ccb option=(occ=1:500)                        (8-2)

trace stor  pattern=fefe address=3aa8:3aa9  output=greg                  (9-1)
trace stor  area=all starea=f3 address=4d13bc:4d13bc output=greg          (9-2)
trace stor  jobname=abcd [stjnam=abcd] address=400abc:400abd
output=greg option=sup                                                   (9-3)
trace stor  area=f3 stdspn=space1 address=4a123:4afff                    (9-4)

trace svc=* area=f3 address=0:* output=greg                              (10-1)
trace svc=(11 1b) area=f7 output=(greg dump reg=a:100)                   (10-2)
```

Figure 20. TRACE command examples

1. Branch trace

   The branch trace monitors successful branch instructions. SDAID displays all branch instructions which transfer control to an address which is contained in the tracing range. 'Successful' branch instruction means that the branching conditions are fulfilled and the processor branches to the specified address. Conditional branch instructions where the branching condition is not met are not traced.

   - 1-1 this command traces instructions with a branch target address in the F3 partition. Since the parameter ADDRESS is omitted, SDAID traces only the area from partition start address to partition end address (default: offset=0:*). The JCL routines are not traced (OPTION=NOJCL). SDAID

displays the general registers ("GREG"), and dumps an area of 256 bytes pointed to by register 7.

- 1-2 this command traces instructions branching to the address BA398. Only the first five occurrences of the specified event are traced (option=(occ=1:5)).
- 1-3 this command traces the BG branch instructions with a branch target address within the phase ABCD. Trace area is an address range starting from phase address + x'200' up to the end of the phase.

2. Cancel trace

   The cancel trace provides an event record and output data whenever the main task in a partition terminates normally or abnormally.

3. External interrupt trace

   This command collects trace data on the occurrence of an external interrupt. The parameter EXT=(...) defines one or more external interruption codes. The parameter EXT=0040 specifies that key interruptions are to be traced. The AREA or ADDRESS parameter are not applicable for the external interrupt trace.

4. Instruction trace

   The instruction trace traces all or selected instructions in a specified area. The specification TRACE INST=BRanch traces all branch type instructions, not regarding the branching conditions. Note the difference from the branch trace which only monitors the branches actually taken.

   - 4-1 this command traces instructions with selected operation codes.
   - 4-2 this command traces supervisor routines working for the F3 partition : The specified address range is not part of the user partition. The parameter option=sup is required for an instruction trace monitoring supervisor routines
   - 4-3 this command traces branch-type instructions. The specification TRACE INST=BRanch traces all branch type instructions, not regarding the branching conditions. Note the difference from the branch trace which only monitors the branches actually taken. The tracing range is the partition selected for the execution of the specified POWER job.
   - 4-4 this command traces all EXECUTE instructions in the dynamic space X9.

5. I/O interruption trace

   The IO trace monitors the I/O interruptions for selected or all I/O devices. The IO trace traces virtual devices, too. The parameter OUTPUT=GREG is not applicable for the IO trace.

   - 5-1 this command traces I/O interruptions for the I/O devices 360 and 361. The parameter OUTPUT=(CCWD=500) monitors the TOD clock, the CCB, the CCWs, the input or output data up to a size of (hexadecimal) 500 bytes and the IRB.
   - 5-2 this command traces I/O interruptions for the device 01f. The parameter OUTPUT=CCW shows the TOD clock, the CCB, the CCWs, and the IRB. Occurrence=1:200 means that only the first 200 occurrences of the traced event are recorded.

6. Program check trace

   - 6-1 this command monitors all program checks except page faults occurring in the F3 partition. SDAID displays the general registers, the partition and task related control blocks and dumps the partition.
   - 6-2 this command traces selected program checks (operation and addressing exception) occurring in the partition where the job TEST is executing. The specification AREA=0:* ensures that program checks in the SVA or in the supervisor are traced, too.

7. Program load trace

   The program load trace monitors phase load instructions. The example traces just the load requests for the phase abcd. If you omit the PHase parameter, all load events are traced. SDAID dumps the general registers and the specified address range.

8. Start subchannel trace
   The SSCH trace monitors SSCH instructions for selected or all I/O devices. The SSCH trace traces virtual devices, too.

   - 8-1 this command displays the SSCH instructions for the device 280. The parameter OUTPUT=TOD displays the time-of-day clock.
   - 8-2 this command displays the first 500 SSCH instructions issued in the F3 partition. SDAID displays the time-of-day clock and the CCB

9. Storage alteration trace

   The storage alteration trace writes an event record whenever an instruction alters the contents of a specified storage location.

   The parameters AREA, JOBNAME, and JOBNUM describe the program which alters a storage location ('the source area'). The parameters STAREA, STJNAM, STJNUM describe the program area where the alteration takes place ('the target space'). The parameter STDSPN defines the name of a data space as target space. The parameters ADDRESS, OFFSET, PHASE and LTA define an address range in the target space. If the target space is a data space, then the sub parameters OFFSET, PHASE and LTA are not valid. You may define a storage alteration pattern. SDAID monitors a storage alteration event only if the specified storage area is set to the specified pattern. The length of the pattern must be equal to the length of the interval specified via the ADDRESS parameter.

   - 9-1 this command traces all instructions (AREA=ALL by default) which alter the specified 2-byte field into the pattern X'FEFE'
   - 9-2 this command traces all tasks (AREA=ALL) which alter the byte at location 4d13bc in the F3 partition
   - 9-3 this command traces storage alterations in the partition where the job ABCD executes. SDAID displays the instructions which alter the half word at location 400abc. SDAID traces the program abcd and the supervisor routines invoked by the program abcd (option=sup). The target partition need not be specified. If JOBNAME=abcd is specified, SDAID takes STJNAM=ABCD per default.
   - 9-4 this command traces all instructions in F3 which alter data at locations 4a123:4afff in the data space SPACE1. The supervisor routines working for F3 are not traced (option=sup is not specified).

10. SVC trace

    The SVC trace monitors all or selected supervisor calls in a specified area.

    - 10-1 this command traces all SVCs in F3. Routines in shared areas are traced, too (ADDRESS=0:*).
    - 10-2 this command traces SVCs X'11' and X'1B' in the F7 partition. SVCs outside the partition space are not traced (OFFSET=0:* by default). SDAID dumps the general registers and an address range of (hexadecimal) 100 bytes pointed to by general register 10.

OUTPUT Parameter

The OUTPUT parameter is used to **include additional dump data into the trace record.** Several z/VSE control blocks may be specified directly by their names. Other storage areas may be specified via the operand DUMP. Figure 21 lists the control blocks which can be specified by name.

Control Block Name	Dumped Area
BUffer	Contents of actual SDAID buffer
CCB, CCW, CCWD=nn	I/O command blocks
COMReg	Partition Communications region
CReg	Control Registers
DUMP	see separate description below
FReg	Floating Point Registers
GReg	General Purpose Registers
IOTab	PUBTAB, LUB, ERBLOC, CHANQ
LTA	Logical Transient Area
LOwcore	Processor Storage from 0 to X'300'
PTA	Physical Transient Area
PTAB	Partition related Control Blocks
SUPvr	Supervisor, CREG, and GREG
SYSCom	System Communications Region
TOD	Time-of-Day Clock
TTAB	Task related Control Blocks

Figure 21. SDAID OUTPut parameter

The parameter OUTPUT=(DUMP ...) may be used with the following sub parameters:

- DUMP PARTition [OFFset=off1:off2]
- DUMP PHase [OFFset=off1:off2]
- DUMP ADDRess=add1:add2
- DUMP REGister=reg:length
- DUMP PTR=reg:offset DMP=offset:length

OPTION Parameter

- OPTION = NOJCL
  The option NOJCL is recommended if you want to trace only the application program and drop any events occurring while the job control statements are processed
- OPTION = SUP
  The option SUP is applicable only for the branch trace, the instruction trace, and the storage alteration trace (the so-called PER traces). The option SUP sets the PER bit into the PSW of the application program **and** into the PSW of supervisor routines if they are active for the specified partition. Option=SUP is not applicable if you specify AREA=ALL or AREA=SUP.
- OPTION = (OCCurrence= mm:nn)
  The option OCC=100:500 means that the first 99 events are dropped. If more than 500 events occur, then the surplus is dropped, too.
- OPTION = Halt
  The option Halt stops the processor after the specified event has occurred. SDAID enters a soft wait state with a PSW of 01060000 00000000 00000000 0000EEEE
- OPTION = Terminate
  The option Terminate stops trace data collection. It monitors the specified event on printer or tape and then terminates trace data collection temporarily. The operator should issue the STOPSD or ENDSD command to terminate SDAID finally.

JCL Procedures for Frequently Used SDAID Functions

A trace initialization is easier if you use the available JCL procedures. The z/VSE system provides JCL procedures with reasonable default values for the most often used tracing functions. Each procedure generates a complete SDAID initialization step with an SDAID statement, an OUTDEV statement, one or two TRACE statements and a READY statement. The generated SDAID initialization statements with the chosen default values are displayed on the screen.

If you use the JCL procedures, observe the following restrictions:

- enclose parameters in quotes
  Any parameter which contains a blank, an asterisk, an equal sign, a comma, a colon, or any other special character must be enclosed in quotes.
- no parameter abbreviations
  The JCL procedures do not accept abbreviated parameters, like ADDR or PH. Exceptions are P for printer, T for tape, and BU for buffer.
- command continuation
  If you enter the procedure statement via SYSIN, enter any non-blank character as continuation character in position 72 and start the continuation line in column 16. If you enter the procedure statement via SYSLOG, break the statement anywhere after a comma. Enter a minus sign as continuation character after a comma, and start the continuation line in column 1.

Procedure Examples

Figure 22 on page 35 gives examples of using the JCL procedures.

1. branch trace procedure SDBRANCH
   The POWER job with the name TEST is traced as soon as it executes in a static or dynamic user partition. If the partition has a begin address of 400000, then SDAID traces all branch instructions with a target address between 40051a and 4006ff. Default value: OPTION=NOJCL. SDAID writes the trace records to the tape 280.
2. instruction trace procedure SDINST
   The procedure SDINST has the following default values: INST=*, OUTPUT=GREG, OPTION=NOJCL
   - 2-1 This procedure traces all instructions in the range 41207a:4123B3 in the F3 partition. The trace output is written to the printer 00E.
   - 2-2 This procedure traces instructions with an operation code of d7 or d2 in the F3 partition. The parameter ADDRESS=0:* makes that instructions inside and outside the F3 partition are traced. Supervisor routines invoked by F3 are traced, too (OPTION=SUP).
   - 2-3 This example shows the usage of a wrap-around buffer.
     The parameter buffer=10 defines a wrap-around buffer. SDAID traces F3 instructions and moves them into the wrap buffer. When a program check occurs in the F3 partition, SDAID writes the buffer contents to the tape device.
     The procedure generates the following OUTDEV and TRACE statements:
     outdev buffer=10 tape=281
     trace inst=* area=f3 address=0:* option=sup
     trace pgmc=* area=f3 address=0:* output=buffer
3. I/O trace procedure SDIO
   The procedure SDIO generates an SSCH trace **and** an I/O interrupt trace. The default output parameter for the SSCH trace is OUTPUT=TOD, the default value for the IO trace is OUTPUT=(CCWD=256). The sample procedure monitors the I/O activity of the unit 431. Trace output device is tape unit 281.
4. Program load trace procedure SDLOAD
   This procedure traces all load requests for the phase ABCD occurring in any partition.
5. Program check trace procedure SDPGMC
   The sample procedure traces all program checks occurring in the attention routine or in a system task (AREA=SUP). It does not trace application programs or supervisor routines active for a user partition. The procedure provides the default value PGMC=*.
6. Storage alteration trace procedure SDSTOR
   - 6-1 This procedure monitors all instructions (AREA=ALL) which alter data in the BG partition at address 41a234 and 41a235.
   - 6-2 This procedure monitors all instructions which alter the contents of the full word at address 41b234 to the value x'12FE34ab'.
7. SVC trace
   This procedure traces all GETVIS and FREEVIS SVCs in the F3 partition. The parameter ADDRESS='0:*' ensures that SVA and LTA routines are traced, too.

```
// exec proc=sdbranch,jobname=test,offset='51a:6ff',tape=280              (1)

// exec proc=sdinst,area=f3,address='41207a:4123b3',printer=00e           (2-1)
// exec proc=sdinst,inst='d2 d7',area=f3,address='0:*',-
tape=281,option='sup nojcl'                                             (2-2)
// exec proc=sdinst,area=f3,address='0:*',option=sup,-
tape=281,buffer=10,buffout=pgmc                                         (2-3)

// exec proc=sdio,unit=431,tape=281                                       (3)

// exec proc=sdload,phase=abcd,tape=281                                   (4)

// exec proc=sdpgmc,area=sup,printer=00e                                  (5)

// exec proc=sdstor,area=all,starea=bg,address='41a234:41a235',-
printer=00e                                                             (6-1)
// exec proc=sdstor,pattern=12fe34ab,-
address='41b234:41b237',printer=00e                                     (6-2)

// exec proc=sdsvc,svc='3d 3e',area=f3,address='0:*',tape=281             (7)
```

Figure 22. SDAID JCL Procedures

If SDAID Does Not Produce Output

Sometimes you trace a program in a static or dynamic partition and do not get any trace output. Often the problem is caused by one of the following reasons:

Address parameter missing

You specified a partition trace (e.g. AREA=F7 or JOBNAME=ABCD) and did not specify an address parameter. If the address parameter is missing, only the area between partition start address and partition end address is traced. The default address specification for a partition trace is OFFSET=0:*. Specify AREA=F7 ADDRESS=0:* to trace all storage inside and outside the F7 partition.

OPTION=SUP is missing.

You specified a branch trace, an instruction trace, or a storage alteration trace for an application program (e.g. AREA=BG or JOBNAME=ABCD). You also specified ADDRESS=0:* (all storage) or an address range in the supervisor area. For the mentioned trace types (BR, INST, STOR) you need to specify OPTION=SUP to set the PER bit into the PSW of the supervisor routines when they are active for the specified partition. The examples (4-2) and (9-3) in Figure 20 on page 29 show the usage of the SUP option.

Expected event not recorded when TRACE=BR

Traces of trace type BRANCH are recording **only** for successfully taken branches. Consider the following environment: Two phases are loaded in F5. PHASE=START and PHASE=END. Phase START will call Phase END once. You want to trace some instructions in PHASE START and finally the Branch to call PHASE END. You have the following TRACE statement:

TRACE BR AREA=BG PHASE=START OFFSET=xxxx:yyyy

Due to the PHASE=START parameter the tracing range is limited to phase START within OFFSET=xxxx:yyyy. All branches which will transfer control inside this range are recorded. The call to phase END transfers control outside and is therfore not recorded. The following Trace setups may fulfill your needs

TRACE BR AREA=BG PHASE=START OFFSET=xxxx:yyyyy
TRACE BR AREA=BG PHASE=END OPTION=TERMINATE or

TRACE INST=BR AREA=BG PHASE=START OFFSET=xxxx:yyyy

Incorrect use of a wrap-around buffer
You specified an instruction trace or an SVC trace and specified a wrap buffer via the command **OUTDEV BUFFER=nn PRINTER=cuu.** SDAID fills the wrap buffer with trace data but it does not print the trace data on the specified printer or tape. SDAID do not print the buffer contents automatically, you have to specify it explicitly according to your needs by one of the following possibilities:

1. Trace Buffer on overflow

   If you want to have the buffer printed whenever it is full and before it will be wraped again you have to include a **TRACE BUFFER** statement in your SDAID setup:

   ```
   OUTEV BU=10    PRINTER= cuu
   TRACE SVC=*
   TRACE INST=* AREA=BG ADDR=600100:600500
   TRACE BUFFER
   ```

2. Trace Buffer on event

   If you want to have the buffer printed whenever a specified event occurs you have to include the **OUTPUT=BUFFER** parameter in the trace statement defining this event.

   ```
   OUTDEV  BU=10   TAPE=cuu   PRINTER= cuu
   TRACE INST=*   AREA=BG    ADDR=600000:600500
   TRACE PGMC=01  AREA=BG    OUTPUT=(BUFFER)
   ```

   Now all instructions between storage addresses: 600000 to 600500 are recorded in wrap around mode into the buffer. If now a program-check 01 occurs the buffer will be printed to the specified printer or tape. This use of a buffer is an efficient way to see the history of a failure without the need to walk over a great bulk of traced data.

If SDAID seems to be in a loop
Sometimes you have started a SDAID Per-type trace (BRANCH INSTRUCTION STOR) and the system do not respond any more. Most likely you have given a wide traceing range. Consider the following examples.

```
SDAID
TRACE  INST=* AREA=BG ADDR=500500:500500  (Traced range is one instruction )
  (etc)
ENDSD
SDAID
TRACE  INST=* AREA=BG ADDR=700500:700500  (Traced range is one instruction )
  (etc)
ENDSD
If you combine two traces like this to one SDAID setup.....................
SDAID
TRACE  INST=* AREA=BG ADDR=500500:500500  (Traced range is one instruction )
TRACE  INST=* AREA=BG ADDR=700500:700500  (Traced range is one instruction )
  (etc)
ENDSD
```

Figure 23. Example of SDAID Trace Range Problem

The combined overall traced range is from 500500 to 700500 which are 2.097.152 bytes of storage. This is because we have only one set of control registers and SDAID uses the lowest and the largest values of the cumulative ranges in CR10 and CR11 respectively. Whenever now a instruction is processed within this range, SDAID gains control and has to filter out the unrelevant events itself, which may lead to a large performance degradation.

SDAID Enhancements

TOD Clock

The TOD (time-of-day) clock has been increased to **micro-seconds**.
Old time stamp e.g. TOD = 2003.307 13.49.36.441
New time stamp e.g. TOD = 2003.307 13.49.36.441692.

GETVIS Trace

The Getvis Trace parameter keyword **SUBPOOL=*subpoolname*** may now contain the value **DEFAULT** as well as any valid hexvalue.

SUBPOOL=DEFAULT With this specification all events which are related to the common overall system subpool will be recorded. In other words, all those events who do not have an explicit subpool name in the GETVIS-MACRO invocation.

SUBPOOL=<*hexvalue*> or SUBPOOL=INCL<0021>
subpoolname may now contain hexvalues or a mix of hexvalues and non-hexvalues. The hexpart must be enclosed by <>.

Enhancements are made available via APAR DY46055. For the TOD-Clock enhancement we recommend to apply DY46140 in addition. This APAR prevents the TOD-clock to run backwards under some seldom conditions.

Important VM/CP Functions

The Virtual Machine/Control Program VM/CP supplies monitoring functions which are similar to the monitoring functions on a native processor. It also supplies a TRACE command which allows to specify a variety of trace types.

z/VSE users can use the TRACE command only if the z/VSE system runs under control of z/VM. The TRACE function traces the actions of the virtual machine in interactive mode. The IBM manual 'z/VM CP Commands and Utilities Reference ' describes the CP monitoring and tracing functions in detail. Figure 24 on page 38 shows some important monitoring functions. The left column shows the CP commands. The right column shows the corresponding monitoring function on a native processor. The figure shows also some TRACE examples. Lowercase letters on CP commands show the possible abbreviation.

****** **VM CP Monitoring Commands** ******	
Ipl cuu	LOAD
Ipl cuu CLEAR	LOAD CLEAR
Display v7a25c-7a3ff	display range of virtual storage
Display b51c.80	display an interval of (hex) 80 bytes
Display PswG	display PSW
EXTernal	INTERRUPT function or IRPT Key
STORE STATUS	Store status
SYSTEM RESTART	System restart
****** **VM CP TRACE Command** ******	
TRace BRAnch INTO 6D2A-6D3B	trace branch instructions into specified target interval
TRace Instruction Range 374C2-374CF	trace instructions within specified interval
TRace I/O 180-182	trace I/O devices 180, 181, 182
TRace STore into 412.2	trace storage alterations at addresses 412 and 413
TRace SVC 3D	trace SVCs with code 3D
TRace END ALL	reset all specified traces

Figure 24. VM/CP monitoring facilities

Lock Manager Service Enhancements

The following lock manager enhancements should ease problem determination and minimize effort on questions around error situations in terms of resources or lock file.

Error on Lock File

Message **0T01E ERROR ON LOCK FILE** indicates one of these situations:

1. DLF statement incorrect
2. an unrecoverable I/O error
3. lock file format error
4. lock file logical error

The investigation of type 1, 2 and 3 errors does not necessarily require a standalone dump. Type 4, which is the most frequent one, however needs analysis of the current lock manager data at the point of failure. The system does not stop processing after issuing the message and therefore a dump at a later point might not show the reason of the problem. In order to speed up the error analysis of such situations a method to circumvent the standard procedure

1. take standalone dump
2. mail to IBM
3. analyze a dump

has to be established.

The lock manager writes together with the message **0T01E ERROR ON LOCK FILE** additional data that describes the error situation. This information can be sent in for analysis via E-MAIL. The contents is internal data and can be changed at any time by IBM if necessary and cannot be considered as an interface.

The example below shows an inconsistency between lock table and lock file when an UNLOCK function was performed.

```
F4 0025 0T01E ERROR ON LOCK FILE
F4 0025 LOCK MANAGER EMERGENCY DATA
V00051138  000C0025 0101000F 80000000 0025D9C5 *        Ø     RE*  R00051138
V00051148  E2D6E4D9 C3C560C5 F1F01110 00051148 *SOURCE-E10     ç*  R00051148
F4 0025 LOCKTAB ENTRY
V0004FE70  00000000 00000000 00000000 00000000 *                 *  R0004FE70
V0004FE80  00000000 00000000 0004FE90 00000000 *          Ú°     *  R0004FE80
F4 0025 LOCK FILE DISK BLOCK
V00086198  D3C60000 D9C5E2D6 E4D9C3C5 60C5F1F0 *LF  RESOURCE-E10*  R00086198
V000861A8  11000000 00000000 00000000 00000000 *                 *  R000861A8
V000861B8  00000000 00000000 00000000 00000000 *                 *  R000861B8
V000861C8  00000000 00000000 00000000 00000000 *                 *  R000861C8
  (etc.)
F4 0025
V00086358  00000000 00000000 00000000 00000000 *                 *  R00086358
V00086368  00000000 00000000 00000000 00000000 *                 *  R00086368
V00086378  00000000 00000000 00000000 00000000 *                 *  R00086378
V00086388  00000000 00000000 00000000 00000000 *                 *  R00086388
F4 0025 LOCK MANAGER TRACE AREA
V0004FF78  D3C3D2E3 0004FF88 00050987 00050788 *LCKT   h   g   h*  R0004FF78
F4 0025 LOCK MANAGER TRACE AREA
V0004FF88  3F3F0B00 00902D80 01080000 01000000 *     ° Ø        *  R0004FF88
V0004FF98  E5C3E3E2 F2F2F0E4 D7D30000 0033B160 *VCTS220UPL    £-*  R0004FF98
V0004FFA8  0033B140 00000000 E5C3E3E2 F2F2F0E4 *  £     VCTS220U*  R0004FFA8
V0004FFB8  D7D30000 01800000 0033B180 0033B120 *PL   Ø    £Ø  £ *  R0004FFB8
  (etc.)
F4 0025
V00050748  25250B00 00600B90 01100000 01000000 *     - °        *  R00050748
V00050758  D9C5E2D6 E4D9C3C5 60C5F1F7 0033B520 *RESOURCE-E17  § *  R00050758
V00050768  0033B510 00000000 D9C5E2D6 E4D9C3C5 *  §     RESOURCE*  R00050768
V00050778  60C5F1F7 01900000 00000000 0033B4E0 *-E17 °        fl\*  R00050778
  (etc.)
F4 0025
V00050948  3F3F0104 00902D80 11000000 01000000 *     ° Ø        *  R00050948
V00050958  E5F24000 9042CCC3 C1E70000 00000000 *V2  °âöCAX      *  R00050958
V00050968  000A0000 00000000 00000000 00000000 *                 *  R00050968
V00050978  00007480 00003F00 070C0000 8007146A *  ÈØ        Ø  ]*  R00050978
```

Figure 25. Sample of inconsistency between lock table and lock file

```
Lock file emergency data contains:
2 bytes: TID (low core)
2 bytes: LCKUTID (lock manager TID, for which the service runs)
1 byte : LOCKPARM (already explained)
1 byte : LOKOWORK (work area to compare DTL info)
1 byte : REQWORK (work flag for deadlock test)
1 byte : UNLCKFLG (flag for unlock service)
1 byte : UNLCKFLG (flag for unlock service: 01 activate waiting tasks
                                           02 free owner element
                                           04 free locktab entry
                                           08 activate E1 requestors
                                           10 lock file block modified
1 byte : DSHRFLG (flag for system task: 80 system task is active
                                        40 timer request already set
                                        20 update on lock file required
                                        10 disk drive reserved
2 bytes: LCKCNT (count locked tasks, deadlock check)
2 bytes: TRCOTID (first owner's TID, if the resource is in use)
12bytes: TRCRESN (current resource name)
1 byte : TRCFLG1 (user's DTL option byte 1)
1 byte : TRCFLG2 (user's DTL option byte 2)
4 byte : reserved (ignore contents)
```

Analysis of the Example

The UNLOCK request came for resource RESOURCE-E10 from task 25(F4). Current task is lock task C.

The UNLOCK was performed successfully in the lock table since the locktab entry was cleared.

However this resource was an external one. TRCFLG2 (DTLFLG2) shows X'10', scope=external.

The corresponding lock file block shows 0 as number of entries. It still can be seen that RESOURCE-E10 has been placed in this block before.

The lock manager detects the mismatch between local lock table and lock file and issues the error message.

The cause of such a type of error can be

- There are several VSE systems under one VM and 2 of them are running with the same CPUID.
- The reserve/release mechanism did not work. Software caching did suppress the lock manager's CCW's but did not provide a proper exclusive access handling. Software caching is not done in VSE. VM or vendor products (on VM or VSE) are candidates.

Display Facility

To allow faster diagnosis in case of resources being in use or deadlock situations it is now possible to display the actual locking status of

- the entire lock table
- all locks held by a specified partition (PIK)
- a given resource name
- a set of resource names, specified with name*
- a set of resource names held by a given partition

This feature is available via a new Attention Routine command **LOCK SHOW**. Examples for the usage of the command are

LOCK SHOW	shows **all** currently held locks (resources) of this **z/VSE system**.
LOCK SHOW=F4	shows **all** currently held locks (resources) of **partition F4**.
LOCK SHOW,RESOURCE-E10	shows **who** is currently holding resource **RESOURCE-E10** (if held at all).
LOCK SHOW,VSYSOPEN'00000001	shows **who** is currently holding resource **VSYSOPEN0000001** (if held at all), where VSYSOPEN is interpreted as characters and 00000001 as hex value.
LOCK SHOW,RESOURC*	shows **who** is currently holding resources, starting with the character string **RESOURC**.
LOCK SHOW=F6,RES*	shows, if **partition F6** is currently holding resources, starting with the character string **RES**.

The output shows unformatted internal data and can be changed at any time by IBM if necessary and cannot be considered as an interface.

```
lock show
AR 0025 LOCKTAB ENTRY
V0004F344  003245A0 00000000 D9C5E2D6 E4D9C3C5 *  áff    RESOURCE*  R0004F344
V0004F354  60C5F1F0 11900001 0004F364 00000000 *-E10 °    3Ã    *  R0004F354
AR 0025 OWNER ELEMENT
V003245A0  00000000 002C0000 00011000 00000000 *               *  R0087F5A0
AR 0025 LOCKTAB ENTRY
V0004F364  0004F3E4 00000000 C4E3E2E5 C5C3E3C2 *  3U    DTSVECTB*  R0004F364
V0004F374  40404040 11800001 0004F384 0004F344 *     Ø    3d  3ã*  R0004F374
AR 0025 OWNER ELEMENT
V0004F3E4  00000000 00400000 00011000 00000000 *               *  R0004F3E4
AR 0025 LOCKTAB ENTRY
V0004F384  0004F3F4 00000000 E5C3E3E2 F2F2F000 *  34    VCTS220 *  R0004F384
V0004F394  00000000 04C00000 0004F3A4 0004F364 *     {    3u  3Ã*  R0004F394
AR 0025 OWNER ELEMENT
V0004F3F4  00000000 00230001 00000000 00000000 *               *  R0004F3F4
AR 0025 LOCKTAB ENTRY
V0004F3A4  0004F414 00000000 E5C3E3E2 F2F2F000 *  4     VCTS220 *  R0004F3A4
V0004F3B4  00000001 04C00000 00324020 0004F384 *     {        3d*  R0004F3B4
AR 0025 OWNER ELEMENT
V0004F414  00000000 00230001 00000000 00000000 *               *  R0004F414
  (etc.)
AR 0025 LOCKTAB ENTRY
V00324440  003245E0 00000000 D9C5E2D6 E4D9C3C5 *  á\    RESOURCE*  R0087F440
V00324450  60C5F1F7 01900000 00000000 00324420 *-E17 °        à *  R0087F450
AR 0025 OWNER ELEMENT
V003245E0  00000000 002C0001 00000000 00000000 *               *  R0087F5E0
AR 0015 1I40I  READY
```

Figure 26. LOCK SHOW command sample

Trace Facility

For the same reason a new trace facility for unsuccessful locks (RC<>0) and unlocks is provided. Trace contents can be specified according to the display facility from above. Besides that a command for trace deactivation is available. Traces can be done for

- all tasks and resources
- all resources belonging to a specified partition (PIK)
- a set of resource names, starting with name* for all tasks
- a set of resource names and a specified partition

The new Attention Routine command **LOCK TRACE** controls the various trace options. Examples for the usage of the command are

LOCK TRACE traces **all** unsuccessful locks and unlocks of **all partitions**.

LOCK TRACE=OFF sets the trace facility **off**.

LOCK TRACE=F4 traces **all** unsuccessful locks and unlocks of **partition F4**.

LOCK TRACE,RESOURCE-E10
traces **all** unsuccessful locks and unlocks of **RESOURCE-E10** of **all partitions**.

LOCK TRACE,VSYSOPEN'00000001
traces **all** unsuccessful locks and unlocks of **VSYSOPEN0000001**, where VSYSOPEN is

interpreted as characters and 00000001 as hex value, of **all partitions**.

LOCK TRACE,RESOURC* traces **all** unsuccessful locks and unlocks of resources, starting with the character string **RESOURC** of **all partitions**.

LOCK TRACE=F6,RES* traces **all** unsuccessful locks and unlocks of resources, starting with the character string **RES** of **partition F4**.

The output shows unformatted internal data and can be changed at any time by IBM if necessary and cannot be considered as an interface.

```
lock trace=f4
AR 0015 1I40I  READY
F4 0025 LOCKTAB ENTRY
V0004F344  00324440 00000000 D9C5E2D6 E4D9C3C5 *  à     RESOURCE*  R0004F344
V0004F354  60C5F1F0 11900001 0004F364 00000000 *-E10 °    3À    *  R0004F354
F4 0025 OWNER ELEMENT
V00324440  00000000 002C0000 00011000 00000000 *               *  R0087F440
F4 0025 LOCKTAB ENTRY
V00324160  00324490 00000000 D9C5E2D6 E4D9C3C5 *  à°    RESOURCE*  R0087F160
V00324170  60C5F1F1 11900001 00324420 00324400 *-E11 °    à   à *  R0087F170
F4 0025 OWNER ELEMENT
V00324490  00000000 002C0000 00011000 00000000 *               *  R0087F490
F4 0025 LOCKTAB ENTRY
V00324450  00324480 00324440 D9C5E2D6 E4D9C3C5 *  àØ  à RESOURCE*  R0087F450
V00324460  60C5F1F3 01900000 003244A0 00324420 *-E13 °    àff  à *  R0087F460
F4 0025 OWNER ELEMENT
V00324480  00000000 002C0001 00000000 00000000 *               *  R0087F480
F4 0025 REQUEST ELEMENT
V00324440  00000000 00250000 00000000 00600B18 *             -  *  R0087F440
AR 0015 1I40I  READY
lock trace=off
AR 0015 1I40I  READY
```

Figure 27. LOCK TRACE command sample 1

```
lock trace=f4,resource-e10
AR 0015 1I40I  READY
F4 0025 LOCKTAB ENTRY
V0004F344  003245C0 00000000 D9C5E2D6 E4D9C3C5 *  á{    RESOURCE*  R0004F344
V0004F354  60C5F1F0 11900001 0004F364 00000000 *-E10 °    3À    *  R0004F354
F4 0025 OWNER ELEMENT
V003245C0  00000000 002C0000 00011000 00000000 *               *  R0087F5C0
F4 0025 LOCKTAB ENTRY
V0004F344  0004F404 003245E0 D9C5E2D6 E4D9C3C5 *  4   á\RESOURCE*  R0004F344
V0004F354  60C5F1F0 11900001 0004F364 00000000 *-E10 °    3À    *  R0004F354
F4 0025 OWNER ELEMENT
V0004F404  00000000 002C0000 00011000 00000000 *               *  R0004F404
F4 0025 REQUEST ELEMENT
V003245E0  00000000 00250000 00000000 00600ABE *             - ”*  R0087F5E0
AR 0015 1I40I  READY
lock trace=off
AR 0015 1I40I  READY
```

Figure 28. LOCK TRACE command sample 2

System Date and Time

VM users may skip this chapter, because the date, clock and time zone values of the installation are extracted from VM, if z/VSE is running as a **guest under VM**. It is not necessary to have any of the SET commands discussed below included in the IPL procedure, unless the z/VSE guest machine is supposed to have a different time than the VM host. On a **native z/VSE system**, however, these values have to be initially provided to the system.

The system date and time of an installation and the TOD clock of the machine can be initialized by the following IPL command:

```
SET DATE= ,CLOCK= ,ZONE=
```

The operand CLOCK specifies the **local** time of the installation. The operand ZONE specifies the time difference between the local time and Greenwich mean time. The TOD clock is then calculated and set to Greenwich mean time, which for historical reasons is the geographical reference for all time zones. The time zone is kept separately. Once the TOD clock has been set, it will continue to run, even if the system is reIPLed. The time zone information, however, is lost.

```
SET DATE=10/31/1997,CLOCK=09/00/00,ZONE=WEST/04/00
```

The command above sets the TOD clock to October 31st, 1997, 1 o'clock p.m. GMT (9 o'clock local time + 4 hours difference to GMT). The system time zone is 4 hours west of Greenwich.

The time zone of the installation has to be specified at each IPL. So a SET ZONE= command is required during IPL, unless your installation is supposed to have Greenwich mean time.

```
SET ZONE=WEST/04/00
```

The command above sets the system time zone of the installation which is located 4 hours west of Greenwich. This would be eastern daylight saving time. The TOD clock is expected to be running, and to have the correct GMT standard time value.

If your machine was powered off, and the TOD clock has to be initially set, IPL will prompt the operator to enter a SET command. In this case please enter a SET command in the format SET DATE= ,CLOCK= ,ZONE= , so that the TOD clock can be calculated and set in relation to the local time of the installation. If the IPL procedure contains a SET ZONE= command, and this is processed **after** the operator was prompted for the SET command, it does not affect the TOD clock any more. The local time, however, would be affected, if the ZONE value differs from the one specified on the previously entered SET command.

The switch from standard time to daylight saving time and vice versa requires a change of the IPL procedure twice a year to adapt the SET ZONE value accordingly.

The **daylight saving function** makes the regular modification of the IPL procedure unnecessary. You may replace the SET ZONE command by a set of new commands, and the system will automatically determine the correct local time when IPL'ed. The new IPL commands allow

1. the definition of time zones,

   ```
   SET ZONEDEF,ZONE=WEST/04/00,EDT
   ```

2. the begin specification of standard time and daylight saving time for a couple of years in advance

   ```
   SET ZONEBDY,DATE=04/06/1997,CLOCK=02/00/00,EDT.
   ```

The first command tells the system, that the time zone with the zone id EDT (eastern daylight time) is located 4 hours west of Greenwich, and the second, that eastern daylight time starts at 2 o'clock on April 6, 1997.

Only an IPL is required to switch to the new time provided the IPL procedure has been updated with a set of the new commands as shown in the example below. It is recommended to place the commands behind the last ADD command.

```
SET ZONEDEF,ZONE=WEST/04/00,EDT
SET ZONEDEF,ZONE=WEST/05/00,EST

SET ZONEBDY,DATE=04/06/1997,CLOCK=02/00/00,EDT
SET ZONEBDY,DATE=10/26/1997,CLOCK=02/00/00,EST

SET ZONEBDY,DATE=04/05/1998,CLOCK=02/00/00,EDT
SET ZONEBDY,DATE=10/25/1998,CLOCK=02/00/00,EST

SET ZONEBDY,...
```

If you reIPLed your system, for example, at 3 o'clock eastern daylight time on October 26, 1997, then IPL would assume now, that the local time is eastern standard time.

You may add up to twenty SET ZONEBDY statements, that means for up to ten years in advance.

Note, that an explicit SET DATE= or SET ZONE= command given during the same IPL will overrule any SET ZONEDEF and SET ZONEBDY statements.

You might want to switch the time on a running system. This can be done, however, it is not officially supported for system integrity reasons. Refer to “TIME” on page 83. See also “Effect of Day-Light Saving Time Changes” on page 84 for some more information on daylight saving time setting.

Chapter 2. Internal Attention Routine Commands

There are quite a few internal commands that were mainly developed for IBM's internal use only, but also have been found to be very useful by system operators. The commands will be described in more detail on the following pages.

However, the intend of these commands is NOT to be construed as an interface of any kind and it is subject to change without notice. Therefore any differences which you may encounter or experience on your system, compared to the description given herein can be the result of such changes.

DEBUG

The DEBUG facility consists of a set of tracing hooks placed at various points within the z/VSE system (mainly supervisor) which need to be activated via a z/VSE operator command. The DEBUG facility, once it has been activated, will create trace entries and save them into 31-bit fixed SVA storage in a wrap around mode fashion. This trace buffer is extremely useful and sometimes even required for problem determination. It will definitely help to decrease the problem solution turn-around-time and should therefor by activated whenever you suspect a system failure. The DEBUG facility, if active, could however impact your z/VSE system performance.

The command DEBUG ON [,nnnK] activates a standard set of traces. The trace types defined by default, monitor the dispatcher program, the first level interrupts, and the I/O activity. If the size of the buffer is not explicitly specified (using option nnnK), the system allocates three buffers with a size of 16k each. Note: The buffers remain allocated until DEBUG END. The buffer size may only be specified once with DEBUG ON before DEBUG END is issued.

For tape library support and for FlashCopy additional AOMxx messages are displayed at the console.

The command DEBUG TRACE allows to modify the set of tracing functions Figure 29 on page 48 shows how you activate and modify the DEBUG function.

DEBUG ON [,nnnk]	activate tracing. 'nnnK' is a custom buffer size and may only be specified once before DEBUG END is issued. The default buffer size is 16K. Three buffers will be allocated.
DEBUG id	activate tracing for the specified partition where **id** is the partition SYSLOG-id (BG, F1,F2...,Zn)
DEBUG OFF	stop DEBUG temporarily
DEBUG END	stop tracing and free all allocated buffers
DEBUG	query tracing status. It will display the currently ACTIVE trace points. Any trace point can be selectively enabled or disabled as the following sample will show.
DEBUG TRACE=REGS,TASK	activate REGiSter and TASK-entry trace
DEBUG TRACE=NOINT,NOSIO	deactivate INTerrupt and Start-IO trace
DEBUG TRACE=ALL,NOSVC	activate all traces, except SVC trace
DEBUG TRACE=NONE,DISP	deactivate all traces, then activate DISPatching trace

Figure 29. DEBUG activation and modification commands

DEBUG STOP

The DEBUG address STOP command compares the contents of a specified storage location with a specified pattern. The z/VSE system performs the compare operation whenever one of the DEBUG event occurs, regardless of whether that entry is active or not. It is possible to compare for equal (EQ), not equal (NE), low (LO), or high (HI). When z/VSE detects a match, it enters a hard wait state with PSW = 000A0000 0000EEEE. You use the RESTART feature of the processor's monitoring facilities or the SYSTEM RESTART command when running under VM, to recover from this hard wait state. Figure 30 shows how you set a DEBUG address STOP.

DEBUG STOP,4B504.4,EQ,FE12ABCD	stop if the full word at 4B504 equals the pattern FE12ABCD.
DEBUG STOP,F4,5AC00C.1,NE,00,OR,180.4,HI,0004ABC0	STOP if the byte at 5AC00C within the F4 addressing scope differs from zero, OR STOP if the four bytes at address 180 are greater than 0004ABC0. You can also establish an AND condition if that should be required. The above sample would then read:
DEBUG STOP,F4,5AC00C.1,NE,00,AND,180.4,HI,0004ABC0	the system would now STOP only if **both** of the previously given conditions are true.

Figure 30. DEBUG STOP commands

DEBUG [{P|N}]SHOW[,ALL]

The DEBUG facility eventually fills three trace buffers in wrap-around mode. Whenever an application program terminates abnormally, the DEBUG facility freezes the current buffer and switches to the next buffer (unless switching has been suppressed (TRACE=NOSWCH). The buffer switching ensures that important trace data is not overlaid by entries generated by the dump routines.

The DEBUG SHOW command provides some options to show ALL or part of any of the three DEBUG-areas. The DEBUG SHOW command cause the traced entries to be formatted and displayed. It is possible to display the buffer contents on the screen or on a printer device. The SHOW command displays the most recent entries in the current buffer. The PSHOW command displays the most recent entries from the previous trace buffer and the DEBUG NSHOW command displays the most recent entries from the trace buffer which is going to be used next. This next buffer, if it contains trace data at all, contains the oldest trace entries which are available in the system.

You can also restrict the DEBUG SHOW output to certain event types only. In this case you would specify the event option e.g. **DEBUG SHOW=SIO,INT** to get only the SIO entries PLUS the INT entries to be displayed/printed.

It is recommended that you issue the command DEBUG OFF before you use the DEBUG SHOW command. Not required for PSHOW or NSHOW. Instead of the DEBUG SHOW command you can also use the command 'DUMP DEBUG,cuu' to print the current DEBUG buffer.

DEBUG SHOW[,CUU=cuu]	display the current DEBUG buffer
DEBUG PSHOW[,CUU=cuu]	display the previous DEBUG buffer
DEBUG NSHOW[,CUU=cuu]	display the next DEBUG buffer

Figure 31. DEBUG SHOW commands

SIR

This is the abbreviation for System Information Report and is a z/VSE Attention command that the operator should use to retrieve useful system information. It provides a variety of functions.

SIR HELP

Syntax: **SIR HELP** or **SIR ?**

This command provides HELP information about all the SIR sub-commands, which are currently available on your system.

The command will produce the following similar output.

```
sir ?
AR 0015       SIR COMMAND HELP
AR 0015  SIR  (<RESET|SYS>)          RESET/DISPLAY SYSTEM INFORMATION
AR 0015  SIR  SMF((,VSE)=<ON|OFF|cuu>) SUBSYSTEM MEASUREMENT DATA
AR 0015  SIR  SMF,ZHPF(,CUU)         ZHPF I/O REQUEST STATISTICS
AR 0015  SIR  MON(=<<id|ON(,NOSYM)>|OFF>(option)) MONITORING DATA
AR 0015  SIR  MIH((,cuu)=<nnnnnn|ON|OFF>) DSPLY/ALTER MIH
AR 0015  SIR  VTAPEBUF(=<nnnK|nnM>)  DISPLAY/ALTER VTAPE BUF-SIZE
AR 0015  SIR  LIBR                   DISPLAY LIBRarien INFORMATION
AR 0015  SIR  CHPID(=chpid)          DISPLAY CHPID INFORMATION
AR 0015  SIR  VENDOR                 DISPLAY VENDOR PRODUCT INF
AR 0015  SIR  CRWMSG(=<ON|OFF>)      DSPLY/ALTER CRW MSG-REPORTING
AR 0015  SIR  VMCF(=<ON|OFF>)        DSPLY/ALTER VMCF INTERFACE
AR 0015  SIR  PMRMON(=<ON|OFF>)      PAGE MANAGER MONITORING DATA
AR 0015 1I40I  READY
```

Figure 32. SIR ? command sample

Explanation

This command provides information about the various SIR sub-commands which are currently available on your system. The command has been provided for user information and convenience. It gives you the keywords and options of any sub-command in a condensed form. For a more detailed description of the sub-commands, you would have to refer to the appropriate description within this booklet.

SIR (without any operand)

This command provides general system information that might be interesting for System-Administrators and -Operators as well. All the timing information (if any) within the different output lines adhere to the following format.

```
hhhh:mm:ss.nnn
(hhhh=hours,mm=minutes,ss=seconds and nnn=milliseconds).
```

The command will produce a report, similar to the following:

The output may change slightly, depending on the setup of your z/VSE system and hardware (native/LPAR/under z/VM).

```
sir
 (1.0)  CPUID  VM = 00036F6A20848000           VSE = FF036F6A20848000
 (1.1)  PROCESSOR = IBM 2084-332 02 (16F6A02)  LPAR = SPA   No. = 0003
 (1.2)       CPUs = 0009 (Ded.=0000 Shr.=0009) Cap. =  28%
 (2.0)  VM-SYSTEM = z/VM    5.2.0   (0801)   USERID = VIASYS   VMCF = ON
 (3.0)       CPUs = 0001                      Cap. =  11%
 (4.0)  PROC-MODE = z/Arch(64-BIT)  IPL(200)    06:42:10      06/09/2008
 (5.0)  SYSTEM    = z/VSE           4.2.0 DR7V                 05/08/2008
 (5.5)              VSE/AF          8.2.0       DRIVER-7      04/17/2008
 (5.55)             VSE/POWER       8.2.0       DRIVER 7      04/17/2008
 (6.0)  IPL-PROC  = $IPLESA         JCL-PROC  = $$JCL
 (7.0)  SUPVR     = $$A$SUPI        TURBO-DISPATCHER (66) ACTIVE
 (7.5)                              HARDWARE COMPRESSION ENABLED
 (7.51)                             CAPACITY MEASUREMENT ACTIVE
 (7.52)                             PAV SUPPORT ACTIVE
 (7.53)                             ZHPF SUPPORT ACTIVE
 (7.55) SEC. MGR. = BASIC           SECURITY  = ONLINE
 (8.0)  VIRTCPU   = 0000:07:28.804         CP = 0000:01:22.938
 (9.0)  CPU-ADDR. = 0000(IPL)   ACTIVE
(10.0)    ACTIVE  = 0000:00:02.100   WAIT = 0000:20:53.322
(11.0)    PARALLEL= 0000:00:00.675   SPIN = 0000:00:00.000
(12.0)  CPU timings MEASUREMENT INTERVAL    0000:20:55.688
(13.0)  TASKS ATT.= 00015          HIGH-MARK = 00015    MAX = 00163
(14.0)  DYN.PARTS = 00000          HIGH-MARK = 00001    MAX = 00048
(15.0)
(16.0)  COPY-BLKS = 00009          HIGH-MARK = 00035    MAX = 01519
(17.0)  CHANQ USED= 00002          HIGH-MARK = 00012    MAX = 00154
(17.5)  LBL.-SEGM.= 00007          HIGH-MARK = 00007    MAX = 00717
(18.0)  PGIN  TOT.= 0000000001     EXP.AVRGE.= 0000000000/SEC
(19.0)  PGOUT TOT.= 0000000001
(20.0)        UNC.= 0000000001     EXP.AVRGE.= 0000000000/SEC
(21.0)        PRE = 0000000000     EXP.AVRGE.= 0000000000/SEC
(22.0)  LOCKS EXT.= 0000000612     LOCKS INT.= 0000005981
(23.0)       FAIL = 0000000014          FAIL = 0000000069
(24.0)  LOCK I/O  = 0000000000     LOCK WRITE= 0000000000
AR 0015 1I40I  READY
```

Figure 33. SIR (without any operand) command sample

Explanation

(1.0) CPUID VM = 00036F6A20848000 VSE = FF036F6A20848000

This line contains the CPUID composed of the CPU Identification Number along with the Version Code followed by the Model Number your native system is running on. If you are running under VM, it also gives you the VIRTUAL CPUID (field VSE) that has been defined via the SET CPUID command under VM (appended with the real model number). In the above example the native processor has a serial number of '036F6A' and is a 2084 processor with version code '00'.

(1.1) PROCESSOR = IBM 2084-332 02 (16F6A02) LPAR = SPA No. = 0003

This line contains processor your z/VSE is running on (incl. serial number) and -if appropriate- additional LPAR information.

(1.2) CPUs = 0009 (Ded.=0000 Shr.=0009) Cap. = 28%

This line displays the configured CPU count which is further subdivided into the dedicated CPU (Ded.) and shared CPU (Shr.) counts and the Logical-Partition Capability Adjustment Factor (Cap.). The Logical-Partition Capability Adjustment Factor specifies the amount of the underlying level-1-configuration CPU capability that is allowed to be used for this level-2 configuration by the LPAR hypervisor.

It represents the maximum capacity available to the LPAR, compared to the capacity of the CEC based on the CEC's total number of shared CPs. The factor is displayed as a percentage value.

The factor calculates as the minimum of the following:

1. # online shared CPs in LPAR / # shared CPs in CEC.
2. LPAR Weight / Total Weight, if Initial Capping is enabled for the LPAR. The Total Weight is the sum of the weights of the active LPARs.
3. Absolute Capping value of the LPAR / # shared CPs in CEC, if an Absolute Capping value is specified for the LPAR.
4. Absolute Capping value of the LPAR group / # shared CPs in CEC, if an Absolute Capping value is specified for the LPAR group to which the LPAR belongs to.

Example: LPAR has 2 online shared CPs and an LPAR Weight of 90, CEC has 10 shared CPs, Total Weight of all active LPARs is 600.
Without any capping: Cap. = 20% (2 / 10)
With Initial Capping enabled: Cap. = 15% (90 / 600)
With Absolute Capping of the LPAR set to 1.00: Cap. = 10% (1.00 / 10)
With Absolute Capping of the LPAR Group set to 0.50: Cap. = 05% (0.50 / 10)

(2.0) VM-SYSTEM = z/VM 5.2.0 (0801) USERID = VIASYS VMCF = ON

This line is OPTIONAL and is only present if you are running under z/VM. It gives information regarding the version of VM you are running, the service level and the z/VM userid your z/VSE is running in. VMCF will show you if your z/VSE is enabled for the “Virtual Machine Communication Facility (VMCF).”

(3.0) CPUs = 0001 Cap. = 11%

This line is OPTIONAL and is only present if you are running under z/VM. It displays the number of CPUs configured for the virtual machine. 'Cap' denotes here the 'Virtual-Machine Capability Adjustment Factor' , which specifies the amount of the underlying level-1-, level-2-, or level-3-configuration capability that is allowed to be used for this level-3 configuration by the virtualmachine control program. This means the maximum ity of the VM compared with the capacity of the underlying CPU configuration, i.e. #cpus in VM / #cpus of underlying configuration. The factor is displayed as a percentage value.

(4.0) PROC-MODE = z/Arch(64-BIT) IPL(200) 06:42:10 06/09/2008

`PROC-MODE` specifies the processors mode of operation.
`IPL` specifies the cuu (e.g 200) that has been used to get your z/VSE system started. Appended is the time and date and OPTIONALLY the time zone of when the z/VSE system was IPL'ed, assuming the time zone is known to the system (defined via the ZONEBDY command).

(5.0) SYSTEM = z/VSE 4.2.0 05/08/2008

This line gives you detailed information about the System refresh level that you are currently running. This sample indicates that it was the z/VSE 4.2.0 level from 05/08/2008.

(5.5) VSE/AF 8.2.0 DY46999 04/17/2008

This line gives you detailed information about the service level of a base component of your z/VSE system. It contains information about the last APAR which was applied to the given component together with the information of when the fix for that APAR had been certified. That date, as such, is an indication of how current your service level for that component is.

This information is extremely useful when you have to place a service problem call (PMR), which, if it is to happen, will enable 1st- and 2nd-Level service support groups to search IBM's data base for possible known problems.

This sample line indicates that the last APAR fix that had been applied to the AF (Advanced Function) component was DY46999 which had been certified on the 04th of April 2008. (Just for those that want to argue: Since a PTF-number is automatically assigned AFTER an APAR has been CLOSED, there is no way to provide the PTF-number instead of the APAR number).

(5.55) VSE/POWER 8.2.0 DY47004 04/17/2008

This is another sample (see the description 5.5 above) which contains information about the last APAR fix which had been applied to the VSE/POWER component.

(6.0) IPL-PROC = $IPLESA JCL-PROC = $$JCL

IPL-PROC specifies the IPL-procedure was chosen by either the system (based on specifications in the $ASIPROC or based on defaults) or by the operator. The named procedure was executed to get your system started.

JCL-PROC specifies the JCL-procedure that had been chosen by either the system (based on specifications in the $ASIPROC or based on defaults) or by the operator. The named procedure was executed to get your partitions started.

(7.0) SUPVR = $$A$SUPI TURBO-DISPATCHER (66) ACTIVE

SUPVR names the supervisor that had been loaded to control your z/VSE system. Appended to the supervisor name you may find information about the turbo dispatcher. The number enclosed in parenthesis (e.g. (66) in our sample) is a hexadecimal number which identifies the Turbo-Dispatcher level. This number will be incremented whenever need arises (which could be due to errors, or due to special services that had eventually been requested by vendors or other sources).

(7.5) HARDWARE COMPRESSION ENABLED

This line is OPTIONAL and gives information on whether the processor does provide the compression facilities (as is the case in our example), or if the software is providing this compression facility.

(7.51) CAPACITY MEASUREMENT ACTIVE

This line is OPTIONAL and gives information on whether the capacity measurement tool is active to allow participation in subcapacity pricing.

(7.52) PAV SUPPORT ACTIVE

This line is OPTIONAL and gives information on whether the z/VSE Parallel Accesss Volume (PAV) support is set active on your z/VSE system.

(7.53) ZHPF SUPPORT ACTIVE

This line is OPTIONAL and gives information on whether the support for zHPF (High Performance FICON for z Systems) is set active on your z/VSE system.

(7.55) SEC. MGR. = BASIC SECURITY = ONLINE

This line contains information about the Security-Manager program that had been chosen (SYS command EMS=name) or invoked by default (BASIC) and its current state.

SEC. MGR Names the security manager that had been requested or selected. When a failure has been recognized during security manager bringup, the name will be set to RECOVER indicating that the system is running WITHOUT a security manager.

BASIC is the synonym for the IBM supplied security manager which will always be loaded when no ESM had been specified.

SECURITY Informs you about the current state of the security manager. The different states are:

INIT FAILED indicates that the initialization of the security manager failed.
INITIALIZING indicates that the initialization of the security manager is ongoing and has not yet been completed.
(IN)ACTIVE indicates that the process of activating the security manager has either not yet been started (INACTIVE) or completed (ACTIVE).
ONLINE and BATCH indicate that the BSM transaction security together with the sign-on security are active. The optional parameter “and BATCH” will only be appended if batch security has been specified in the SYS command (SEC=YES).

(8.0) VIRTCPU = 0000:07:28.804 CP = 0000:01:22.938

This line is OPTIONAL and only present when running under VM. It gives you information about the real processor time that your virtual system has consumed until now.

VIRTCPU is the total virtual time consumed by the virtual guest system and
CP is that portion of the total virtual time (VIRTCPU) that CP had spent specifically on behalf of the virtual CPU. In our sample the total time consumed was 7 minutes, 28 seconds and 804 milliseconds whereof 1 minutes and 22.938 seconds where consumed by CP itself.

(9.0) CPU-ADDR. = 0000(IPL) ACTIVE

This line is OPTIONAL and only present if you are running the TURBO DISPATCHER and shows the different CPU addresses and their state of operation. This and the following lines (ref:10.0 and 11.0) will be repeated for each processor that exists in your system. The information subsequent to these line(s) (ref: 10.0 and 11.0) always applies to the CPU address (CPU-ADDR) named last.

CPU-ADDR is the CPU address that the subsequent information (ref 10.0 and ref 11.0) is associated to. Appended to the CPU address you find information about the current state of this processor. In this example the processor is ACTIVE. Other states could be INACTIVE, STOPPED, IN ERROR etc.. The information (IPL) is appended to one and only one processor out of a range of processors that constitute an MP system and it indicates that processor, that had been IPL'ed and it also indicates that this is the processor that can't be STOPPED by the operator.

(10.0) ACTIVE = 0000:00:02.100 WAIT = 0000:20:53.322

This line is OPTIONAL and only present if you are running the TURBO DISPATCHER and it is related to the processor address immediately preceding this line and it gives you information about the processors CPU time consumption. In our sample it indicates that the processor with the CPU address 0000 has ACTIVEly consumed 2 seconds and 100 milli-seconds and it had been

in a WAITing state for 20 minutes 53 seconds and 322 milli-seconds. The active time does INCLUDE the SPIN time (ref:11.0).

(11.0) PARALLEL= 0000:00:00.675 SPIN = 0000:00:00.000

This line is OPTIONAL and only present if you are running the TURBO DISPATCHER and it is related to the last preceding processor address and gives information about the work-mix that this processor was executing.

PARALLEL is the time that this processor executed work where other processors could work concurrently. The higher this value, the better the utilization of Multiple Processors (MP). In our sample it indicates that the work-mix on this processor was such that about 30% (0.675 seconds out of 2.100 seconds ACTIVE time (ref 10.0) could be executed in parallel.

SPIN is the time that this processor spent in a very tight loop waiting for system resources occupied by other processors within the MP-system to be freed. In our sample processor 0000 spent 0 milliseconds in this tight loop.

(12.0) CPU timings MEASUREMENT INTERVAL 0000:20:55.688

This line is OPTIONAL and only present if you are running the TURBO DISPATCHER and it defines the time that has elapsed since the system was either IPLed, or since the measurement values have been reset (SYSDEF TD,.....). All the values from the preceding lines (ref:10.0 and 11.0) are related to this measuring interval. In our sample the processor activity (for all the processors mentioned before) was measured for 20 minutes and 55.688 seconds.

(13.0) TASKS ATT.= 00015 HIGH-MARK = 00018 MAX = 00163

This line informs you about the number of tasks that are, have maximal been, and could ever be ATTACHED concurrently.

TASK ATT. is the number of tasks currently ATTACHED.

HIGH-MARK is the maximum number of task that had ever been attached concurrently since IPL.

MAX is the maximum number of task that can ever be attached concurrently. It is the smaller value of: 256-32-NPARTS and 208 In the above example we have currently 15 tasks ATTACHED; we had, at a certain point in time, a maximum of 18 task(s) ATTACHED and we could attach a maximum of up to 163 tasks.

(14.0) DYN.PARTS = 00000 HIGH-MARK = 00001 MAX = 00048

This line informs you about the number of DYNAMIC PARTITIONS that are, have maximal been, or could ever be ALLOCATED concurrently.

DYN.PARTS is the number of DYNAMIC PARTITIONS currently active.

HIGH-MARK is the maximum number of DYNAMIC PARTITIONS that had been allocated concurrently since IPL.

MAX is the maximum number of DYNAMIC PARTITIONS that can be allocated concurrently.

(15.0)

This line is left blank intentionally

(16.0) COPY-BLKS = 00009 HIGH-MARK = 00035 MAX = 01519

COPY-BLKS is the number of COPY-BLOCKS (required to properly perform CCW-Translation) that currently are, have ever been (since last IPL), or can maximal ever be used concurrently by the currently running system.
In our sample we have currently 9 copy-blocks in use.

HIGH-MARK gives the maximum number of copy-blocks that had ever been used concurrently (since the last IPL).
In our sample the maximum number of copy-block that had been concurrently in use at a certain point in time (since last IPL) was 35.

MAX specifies the number of copy-blocks that the currently running system has allocated. This value can never be exceeded without IPL'ing the system.

Note: If HIGH-MARK ever gets close to the MAX-value, you should consider adding more BUFSIZE entries to prevent running out of them in the future, which could lead to performance degradation.
Special care should be taken of this value if you consider to activate VTAM 31-bit support or have this support active !
If the HIGH-MARK never gets even close to the MAX-value, then it would be an indication to run with a smaller number of BUFSIZE entries (to save 72-bytes per entry) when you perform a new IPL.

(17.0) CHANQ USED= 00002 HIGH-MARK = 00012 MAX = 00154

CHANQ USED is the number of CHANnel Queue entries (one entry is required per I/O operation) that currently are, have ever been (since last IPL), or can maximal ever be used concurrently by the currently running system. In our sample we have currently 2 channel-queue entries in use. Having a channel queue entry in use does not necessarily indicate that an I/O operation is currently ongoing.

HIGH-MARK gives the maximum number of channel queue entries that had ever been used concurrently (since the last IPL). In our sample the maximum number of copy-block that had been concurrently in use at a certain point in time (since last IPL) was 12.

MAX specifies the number of channel queue entries that the currently running system has allocated. This value can never be exceeded without IPL'ing the system.

Note: If HIGH-MARK ever gets close to the MAX value, you should consider adding more CHANQ entries to prevent running out of them in the future , which could lead to performance degradation. If it never gets even close to the MAX value, then it would be an indication to try to run with a smaller number of CHANQ entries (to save 32-bytes per entry) when you perform a new IPL. A user specified number may, however, be overruled by IPL depending on the number and type of I/O devices ADDed.

(17.5) LBL.-SEGM.= 00007 HIGH-MARK = 00007 MAX = 00717
this line is optional and only present if the label area is on a virtual FBA disk (VDISK) and it contains the number of label segments (2K each) that are currently being used, that have ever been used (since last IPL) or which can maximal be used by the currently running system. In our sample we have currently 7 Label-segments in use. The number of label-segments is not telling anything about the number of labels being used, simply because part of that area is being used internally and because the labels are variable in length.

HIGH-MARK gives the maximum number of label-segments that had ever been used concurrently (since the last IPL). In our sample the maximum number of label-segments that had been concurrently in use at a certain point in time (since last IPL) was 7.

MAX specifies the number of label-segments that the currently running system has allocated. This value can never be exceeded without IPL'ing the system.

Note: If HIGH-MARK ever gets close to the MAX value, you should consider increasing your label area to prevent running out of label-segments which could result in problems during Job execution. If it never gets even close to the MAX value, then it would be an indication to try to run with a smaller label-area when you perform a new IPL.

The next lines (ref: 18.0 through 21.0) are OPTIONAL and only present if you are running a system with a PDS (Page Data Set) having been allocated during IPL.

(18.0) PGIN TOT.= 0000000001 EXP.AVRGE.= 0000000000/SEC

This line informs you about the PAGE-IN activity since you have IPL'ed the system. It is an indication of the total PAGE I/O activity that was (PGIN TOT) or is (EXP.AVRGE.) ongoing.

PGIN TOT. is the total number of PAGE-IN requests that have been performed since the last IPL.

EXP.AVRGE is the EXPonential AVeRaGE PAGE-IN rate measured in requests per second. Exponential Average means that the LATEST PAGE-IN activity, measured during the latest measuring interval (which is the time consumption between a predefined number of PAGE-IN requests, which is processor dependant) is taken into account with an equal weight, compared to the OLD exponential average, in calculating the NEW exponential average.

The formula is: *((LATEST-PAGE-IN/second)+(OLD-EXP.AVRGE))/2* to illustrate this, assume an: EXP.AVRGE.= 000000038/SEC and assume a time difference from 200 milli-seconds between a 1st and the 10th PAGE-IN request (assuming a processor constant of 10), then the NEW EXP.AVRGE would calculate to:

```
LATEST PAGE-IN          200ms/10pgin = 50pgin/second
NEW EXP.AVRGE          (50pgin/sec + 38pgin/sec) / 2 = 44
and thus our line would say:              EXP.AVRGE.= 44
```

(19.0) PGOUT TOT.= 0000000001

This line informs you about the PAGE-OUT activity since you have IPL'ed the system.

PGOUT TOT. is the total number of PAGE-OUT requests that have been performed since the last IPL. This line is the accumulated value of the next two lines
(ref:20.0 and 21.0).

(20.0) UNC.= 0000000001 EXP.AVRGE.= 0000000000/SEC

UNC. gives the number of UNCONDITIONAL PAGE-OUT requests which have either been explicitly requested, or which have been initiated by the PAGE MANAGER to free one or more PAGE FRAMEs.

EXP.AVRGE is the EXPonential AVeRaGE UNC.PGOUT rate measured in requests per second. Exponential Average means that the LATEST UNC.PGOUT activity, measured during the latest measuring interval (which is the time consumption between a predefined number of

PAGE-IN requests, which is processor dependant) is taken into account with an equal weight, compared to the OLD exponential average, in calculating the NEW exponential average.

The formula is: *(LATEST-UNC.PGOUT/second)+(OLD-EXP.AVRGE)/2* A detailed sample that you may want to refer to is given above (ref:18.0).

(21.0) PRE = 0000000000 EXP.AVRGE.= 0000000000/SEC

`PRE` gives the number of PRE PAGE-OUT requests which have been initiated by the PAGE MANAGER to ensure PAGE FRAME availability when a PAGE-IN request should be received. These values might serve as an indication of how often pages are being referenced by the system. Pages not being referenced any more, will be subject to destaging (PRE-PAGE-OUT) if need arises.

`EXP.AVRGE` is the EXPonential AVeRaGE PRE.PGOUT rate measured in requests per second. Exponential Average means that the LATEST PRE.PGOUT activity, measured during the latest measuring interval (which is the time consumption between a predefined number of PAGE-IN requests, which is processor dependant) is taken into account with an equal weight, compared to the OLD exponential average, in calculating the NEW exponential average.

The formula is: *(LATEST-PRE.PGOUT/second)+(OLD-EXP.AVRGE)/2* A detailed sample that you may want to refer to is given above (ref:18.0).

The next lines (ref: 22.0 through 24.0) give you some information about the LOCKTAB and/or LOCKFILE (DLF) usage and it tells you about conflicts that have been encountered due to attempts to access certain resources concurrently.

(22.0) LOCKS EXT.= 0000000612 LOCKS INT.= 0000005981

This line contains the total number of LOCK (USE) requests that have been issued since the last IPL.

`LOCKS EXT.` is the total number of EXTERNAL locks that had been requested. EXTERNAL means that the requestor has requested a resource in a SHARED environment for controlled access. It DOES NOT indicate that you are RUNNING a shared environment (e.g. the user could have requested an EXTERNAL LOCK, but the system is running WITHOUT a DLF command (see ref:24.0)). In the above sample there has been a total of 612 LOCK request to guarantee controlled access by other processors that do have access to the specified resource.

`LOCKS INT.` is the total number of INTERNAL locks that had been requested. INTERNAL means that the requestor has requested a resource in his own system to be locked for controlled access. Internal LOCKS are those for which controlled access by other tasks within this system is granted. In our sample we had 5981 LOCK (USE) requests with the attribute INTERNAL (grant controlled access by other tasks).

(23.0) FAIL = 0000000014 FAIL = 0000000069

This line contains the total number of UNSUCCESSFUL LOCK attempts either EXTERNAL (left column) or INTERNAL (right column)

`FAIL` is the total number of LOCK requests that had been unsuccessful due to access constrains with other requestors. In the above example, 14 EXTERNAL LOCK requests (out of 612 (ref:22.0)) had been unsuccessful and as such had to be re-issued. Similar, 69 out of

5981 INTERNAL LOCK requests had been unsuccessful and would have had to be re-issued.

(24.0) LOCK I/O = 0000000000 LOCK WRITE= 0000000000
This last line gives information about the LOCKFILE I/O accesses.

`LOCK I/O` is the total number of I/O operations, whereby the LOCK request itself might have been success-or unsuccessful, that had been issued to the LOCKFILE.

`LOCK WRITE` is the total number of I/O operations where an external lock was either obtained, or freed. The greater the difference between these two values is, the more unbalanced is the workload among the different processors that constitute the SHARED environment. The fact that in our sample the LOCK I/O is 00000000, indicates that we where running a NON-SHARED system.

SIR SYS

Syntax: **SIR SYS**

This command allows to retrieve only part of the data that SIR would have provided. The output will be limited to the static information only (ref: 1.0 through ref: 7.55)

SIR RESET

Syntax: **SIR RESET**

This command allows to reset the dynamic counters which are subject to the SIR command output (ref: 8.0 through 24.0) back to zero and to start another counting cycle.

SIR SMF

Syntax: **SIR SMF[={ON|OFF}][,VSE][,ZHPF][,cuu]**

This command allows to retrieve I/O data maintained by either the channel subsystem, when the channel subsystem measurement facility exists, or by the I/O supervisor itself. It is mainly intended to assist in finding I/O performance problems. Since the channel subsystem does maintain these counters without posting an overflow condition, it is recommended to not run this activity report unlimited in order to prevent confusing results.

SMF is the keyword indicating that Subsystem Measurement Facility data is requested and, if specified without any further options, will retrieve that data, assuming that SMF had been activated before.

- **ON** indicates that you want to activate I/O performance measurements functions in z/VSE and to also activate the Channel Subsystem Measurement Facility (if available). This is a prerequisite to retrieve SMF data.
- **OFF** indicates that you want to deactivate the I/O performance measurement functions in z/VSE as well as in the Channel Subsystem.
- **VSE** indicates that you want the I/O counters as maintained by the VSE I/O supervisor to be displayed assuming that SMF is already active. Differences that you may encounter between the counters maintained by z/VSE and those figures as retrieved from the Channel Subsystem

could eventually be caused by either VM or other programs which may have intercepted the appropriate z/VSE code. So the difference is most likely the time that had been spend in VM or vendor program products or even both.

ZHPF indicates that you want the I/O counters to be displayed as maintained by the VSE I/O supervisor for z/HPF enabled devices. These counters are available and will be displayed for PAV and non-PAV devices when the z/HPF support has been activated.

cuu indicates that you want the data for just one single device which had been identified by its device-Id (cuu).

The following page is some extracts from a console log which should help the reader to get a better understanding of the function of the SIR SMF command.

```
sir smf
  (1.0) SUBSYSTEM MEASUREMENT FACILITY IS INACTIVE
sir smf=on
  (2.0) SUBSYSTEM MEASUREMENT FACILITY HAS BEEN ACTIVATED
  (3.0) NO SUBSYSTEM MEASUREMENT DATA YET AVAILABLE
        -------here some I/O activity was forced-----------
AR 0015 CUU  CODE DEV.-TYP   VOLID  USAGE   SHARED    STATUS    CAPACITY
AR 0015 B3B  6E   2105-000  XSAMC2  UNUSED                          3339 CYL
AR 0015 B3C  6E   2105-000  XSAM02  UNUSED                          3339 CYL
AR 0015 B3D  6E   2105-000  XSAM04  UNUSED                          3339 CYL
AR 0015 B3E  6E   2105-000  XSAM06  UNUSED                          3339 CYL
AR 0015 B3F  6E   2105-000  XSAM08  UNUSED                          3339 CYL
AR 0015 B44  6E   2105-000  XSAMC1  UNUSED                          3339 CYL
AR 0015 B45  6E   2105-000  XSAM01  UNUSED                          3339 CYL
AR 0015 B46  6E   2105-000  XSAM03  UNUSED                          3339 CYL
AR 0015 B47  6E   2105-000  XSAM05  UNUSED                          3339 CYL
AR 0015 B48  6E   2105-000  XSAM07  UNUSED                          3339 CYL
AR 0015 B51  6E   2105-000  VSAMC1  UNUSED                          3339 CYL
AR 0015 B52  6E   2105-000  VSAM01  UNUSED                          3339 CYL
AR 0015 B53  6E   2105-000  VSAM03  UNUSED                          3339 CYL
AR 0015 B54  6E   2105-000  VSAM05  UNUSED                          3339 CYL
AR 0015 B55  6E   2105-000  VSAM07  UNUSED                          3339 CYL
AR 0015 B56  6E   2105-000  VSAMC2  UNUSED                          3339 CYL
AR 0015 B57  6E   2105-000  VSAM02  UNUSED                          3339 CYL
AR 0015 B58  6E   2105-000  VSAM04  UNUSED                          3339 CYL
AR 0015 B59  6E   2105-000  VSAM06  UNUSED                          3339 CYL
AR 0015 B5A  6E   2105-000  VSAM08  UNUSED                          3339 CYL
AR 0015 F15  6E   2105-000  DOSRES  USED                            3339 CYL
AR 0015 F16  6E   2105-000  SYSWK1  USED                            3339 CYL
sir smf
  (4.0) DEVICE  I/O-CNT     QUEUED      CONNECT     DISCONN     TOTAL
  (4.1)                   msec/SSCH   msec/SSCH   msec/SSCH   msec/SSCH
  (5.0)
  (6.0)  B3B        2426       0.353       0.366       0.000       0.719
  (6.0)  B3C       18014       0.341       0.634       0.010       0.986
  (6.0)  B3D       18014       0.367       0.640       0.011       1.019
  (6.0)  B3E       18006       0.355       0.635       0.007       0.998
  (6.0)  B3F       18014       0.357       0.622       0.008       0.988
  (6.0)  B44        2426       0.296       0.304       0.007       0.608
  (6.0)  B45       18014       0.322       0.580       0.013       0.915
  (6.0)  B46       18014       0.327       0.594       0.012       0.934
  (6.0)  B47       18006       0.343       0.596       0.008       0.947
  (6.0)  B48       18014       0.359       0.629       0.009       0.997
  (6.0)  B51        2426       0.321       0.347       0.002       0.671
  (6.0)  B52       18014       0.340       0.618       0.010       0.968
  (6.0)  B53       18014       0.336       0.631       0.007       0.975
  (6.0)  B54       18006       0.357       0.623       0.010       0.991
  (6.0)  B55       18014       0.325       0.623       0.039       0.988
  (6.0)  B56        2426       0.329       0.364       0.000       0.693
  (6.0)  B57       18014       0.369       0.654       0.008       1.031
  (6.0)  B58       18012       0.355       0.607       0.013       0.977
  (6.0)  B59       18006       0.293       0.606       0.008       0.908
  (6.0)  B5A       18014       0.350       0.619       0.009       0.979
  (6.0)  F15       18102       0.759       0.678       0.000       1.439
  (6.0)  F16       16469       0.481       0.634       0.001       1.116
  (6.0) 1I40I  READY
        ----------------------------------------------------
```

Figure 34 (Part 1 of 2). SIR SMF command sample

```
sir smf=b51
 (10.0) TIMINGS FOR B51 BASED ON       2426 I/O INSTRUCTION
 (11.0)
 (12.0)   QUEUED      PENDING     CONNECT     DISCONN    DEV.BUSY      TOTAL
 (13.0)  msec/SSCH   msec/SSCH   msec/SSCH   msec/SSCH   msec/SSCH   msec/SSCH
 (14.0)      0.008       0.313       0.347       0.002       0.000       0.671
 (15.0) 1I40I  READY
        ---------------------------------------------------
sir smf,vse,b51
 (10.0) TIMINGS FOR B51 BASED ON       2426 I/O INSTRUCTION
 (10.5) MAXIMUM I/O QUEUE   2
 (11.0)
 (12.0)   QUEUED      PENDING     CONNECT     DISCONN    DEV.BUSY      TOTAL
 (13.0)  msec/SSCH   msec/SSCH   msec/SSCH   msec/SSCH   msec/SSCH   msec/SSCH
 (14.0)      0.008       0.000       0.949       0.000       0.000       0.958
 (15.0) 1I40I  READY
        ---------------------------------------------------
sir smf,zhpf,178
(16.0)   AR 0015 ZHPF I/O REQUEST STATISTICS
(16.1)   AR 0015   CUU          OVERALL       ZHPF   REJECTED
(16.2)   AR 0015   178 BASE          29          0          0
(16.3)   AR 0015  91FC ALIAS          0          0          0
(16.3)   AR 0015  91FD ALIAS          0          0          0
(16.3)   AR 0015  91FE ALIAS          0          0          0
(16.3)   AR 0015  91FF ALIAS       1109        865          0
        ---------------------------------------------------
sir smf=off
AR 0015 1I40I  READY
sir smf
  (1.0) SUBSYSTEM MEASUREMENT FACILITY IS INACTIVE
```

Figure 34 (Part 2 of 2). SIR SMF command sample

Explanation

(1.0) SUBSYSTEM MEASUREMENT FACILITY IS INACTIVE

This message is telling you that the Subsystem Measurement Facility (SMF) is inactive, or has been deactivated (SIR SMF=OFF).

The operator would have to issue the SIR SMF=ON command to get the facility initialized properly.

(2.0) SUBSYSTEM MEASUREMENT FACILITY HAS BEEN ACTIVATED

This is the acknowledgement message that the Measurement Facility has now been successfully initialized.

(3.0) NO SUBSYSTEM MEASUREMENT DATA YET AVAILABLE

This message indicates that the Subsystem Measurement Facility is active, but that NO measurement data is yet available. Either no I/O operation has been issued since the Measurement Facility has been activated, or the device is not being supported by the measurement facility or by the I/O supervisor.

(4.0) DEVICE I/O-CNT QUEUED CONNECT DISCONN TOTAL
(4.1) msec/SSCH msec/SSCH msec/SSCH msec/SSCH

This line is a heading line describing the contents of the vertical fields below the headings subject. Timing values are given in milli-seconds per SSCH (START SUBCHANNEL).

`DEVICE` specifies the cuu address of the device that is being measured.

I/O-CNT specifies the number of I/O operations that had been used in calculating the measurement data in the appropriate line. Watch-out, the I/O-CNT could have wrapped at 65,536 back to zero while the timings will be accumulated for about 153 hours which could cause inconsistent data once the I/O-CNT wraps. To prevent such inconsistent data, it is recommended to run the Suchannel Monitoring Facility only for a certain time and then turn it off via the SIR SMF=OFF command. A subsequent re-activation (SET SMF=ON) will start with all fields reset to zero.
Having z/VSE PAV support active an I/O-CNT of "*" indicates an overflow.

QUEUED this field contains the average time (based on the I/O-CNT) that an I/O request was queued in z/VSE (I/O supervisor). If the PENDING time is not given explicitely (ref: 12.0) then the QUEUED time does also INCLUDE the time the request was PENDING in the Channel Subsystem. An I/O request is queued in z/VSE from the time the I/O request is enqueued up to the point where the Start I/O operation (SSCH) has successfully been initiated. Starting with the successful initiation of the SSCH instruction, the time will be counted as PENDING in Channel Subsystem. A request would be held pending in the Channel Subsystem if e.g. a channel or a Control Unit (CU) is currently busy. The values in this column would thus indicate a device, a channel or a CU (Control Unit) contention.

CONNECT this field contains the average time (based on the I/O-CNT) that a device is logically connected to a channel for purposes of transferring information between it and the Channel Subsystem.

DISCONN this field contains the average time (based on the I/O-CNT) that a device is logically disconnected from the Channel Subsystem while the device is still busy and has not yet presented primary interrupt (Channel End) status. In case a DEV.BUSY time is not outlined explicitly (ref: 12.0), then the DISCONN time does also INCLUDE the DEV.BUSY time which is the time between the primary status (CE) and the secondary device-end status (DE).

TOTAL this field contains the average time (based on the I/O-CNT) of a complete I/O operation and is actually the sum of QUEUED (+PENDING), CONNECT and DISCONN (+DEV.BUSY). An excessive value in this field could be the indication of a device problem.

(6.0) B3B 2426 0.353 0.366 0.000 0.719

This is a sample line telling you that the average time, that had been measured based on 2426 SSCH-instructions, issued to the device with the address B3B, had been 738 micro-seconds in total. From this 0.719 milli-seconds, the request was QUEUED in the z/VSE I/O supervisor and was PENDING in the channel subsystem for about half a milli-second (0.353), was doing data transfer operations for 0.366 milli-seconds and was no executing SEEK/SEARCH type operations (DISCONNected). This sample might be considered a typical sample for CACHED DASD I/O operations with CACHE HITS. The other lines (ref: 6.0) are similar.

EXPLANATION for a single device:

(10.0) TIMINGS FOR B51 BASED ON 2426 I/O INSTRUCTION

This line gives the device-Id (in our sample B51) plus the number of I/O instructions (in the sample there were 2426) that the subsequent measuring data is based on.

(10.5) MAXIMUM I/O QUEUE 2
This line shows the maximum, concurrently queued I/O's on the device.

(11.0)
This line is left blank intentionally

(12.0) QUEUED PENDING CONNECT DISCONN DEV.BUSY TOTAL
This line is a heading line describing the contents of the vertical fields below the headings subject. The heading line as well as the subheading line will be repeated after 16 detail lines to make the console output easier to read. Timing values are given in milli-seconds per SSCH (START SUBCHANNEL).

`QUEUED` this field contains the average time (based on the I/O-CNT) that an I/O request was queued in z/VSE (I/O supervisor). An I/O request is queued in z/VSE from the time the I/O request is enqueued up to the point where the Start I/O operation (SSCH) has successfully been initiated. High values in this field would indicate a device contention.

`PENDING` this field contains the average time (based on the I/O-CNT) that an I/O request was held PENDING in the channel subsystem after the SSCH instruction had been accepted. A request would be held pending in the Channel Subsystem if e.g. a channel or a Control Unit (CU) is currently busy. The values in this column would thus indicate a channel or a CU (Control Unit) contention.

`CONNECT` this field contains the average time (based on the I/O-CNT) that a device is logically connected to a channel for purposes of transferring information between it and the Channel Subsystem.

`DISCONN` this field contains the average time (based on the I/O-CNT) that a device is logically disconnected from the Channel Subsystem while the device is still busy and has not yet presented primary interrupt (Channel End) status. This value does NOT include the time between Channel End (CE) and Device End (DE).

`DEV.BUSY` this field contains the average time (based on the I/O-CNT) that a device is still busy, while the subchannel is idle. Thus, it is the time between Channel End (CE) and Device End (DE).

`TOTAL` this field contains the average time (based on the I/O-CNT) of a complete I/O operation and is actually the sum of QUEUED, PENDING, CONNECT, DISCONN and DEV.BUSY time. It should be used to quickly check if there are any I/O-device problems.

(14.0) 0.008 0.313 0.347 0.002 0.000 0.671
This line (based on 2426 SSCH instruction to device X'b51') is now stating that the I/O operation was queued for 8 micro-seconds in z/VSE, that it took the channel subsystem about 313 micro-seconds to get it initiated successfully and that the data-transfer operation (CONNECT) did last 347 micro-seconds. The time the device was DISCONNected from the channel (for DASD typically the SEEK/SEARCH times) was 2 micro-seconds and there was 0 micro-seconds that the device was busy with the subchannel being idle. This indicates that we did not have split channel- and device end (CE and DE) interrupts. This resulted in a TOTAL of 671 micro-seconds for the whole I/O operation.

The second sample is similar to the first one, except that this time the figures are based on the counters maintained by z/VSE itself.

(16.0) AR 0015 ZHPF I/O REQUEST STATISTICS
I/O statistics for z/HPF are displayed.

16.1) AR 0015 CUU OVERALL ZHPF REJECTED
Heading line for z/HPF I/O statistics.
Column OVERALL shows the total I/O count, column ZHPF shows the number or I/Os that have been converted for z/HPF, and column REJECTED shows the number of z/HPF converted I/Os that have been rejected by the device.

(16.2) AR 0015 178 BASE 29 0 0

(16.3) AR 0015 91FC ALIAS 0 0 0
List of I/O counters per device. For PAV-enabled devices, these counters are displayed for the base device and for each of the alias devices .

SIR MON

Syntax: **SIR MON[={id|ON[,NOSYM]|OFF [,{FAST|COUNTER|BOUND}]**

This command provides monitoring data and is available with the TURBO DISPATCHER only. The information will consist of a certain function and the number of times this function has been executed. The function is either a symbolic name/macro or a hexadecimal value, whereas the execution count is always a decimal value.

MON is the keyword required to retrieve monitoring data.

`id` is the partition SYSLOG-Id which is to be monitored exclusively (BG,F1,F2...,Zn). You can also pass a task id in the form Txx, where xx would be the HEX value of the task that you want to monitor. Please note that before switching to another id, you should issue the **SYSDEF TD,RESETCNT** command to get all counters reset to zero. If specified without any further operands, it will cause ALL monitoring services to be activated (SVC, FAST, COUNTER and BOUND).

`ON` indicates that monitoring is to be started. If specified without any further operands, it will cause ALL monitoring services to be activated (SVC, FAST, COUNTER and BOUND).

`NOSYM` indicates that the function code substitution into symbol names is to be suppressed and that the function code is to be supplied by its hexadecimal value instead.

`OFF` indicates that monitoring is to be stopped. If specified without any further operands, it will cause all monitoring to be stopped. If FAST or BOUND or COUNTER have been specified, monitoring will be deactivated for that special service only. Please note that this option does **NOT** reset the counters. You must use the **SYSDEF TD,RESETCNT** command instead.

`FAST` indicates, that, in addition to the supervisor calls (SVCs, also the FAST-SVC calls (SVC-6B) be explicitly monitored by their function code.

`COUNTER` indicates, that, in addition to the supervisor calls (SVCs), also the Turbo Dispatcher (TD) tuning counters be explicitly monitored. These counters are intended for internal use only to assist in performance problem analysis.

`BOUND` indicates, that, in addition to the supervisor calls (SVCs), also the WAIT conditions (Bound states) be explicitly monitored.

To retrieve the monitoring data, you just need to issue the `SIR MON` command (without any further operand). The command will produce similar output as shown in the sample below. In the SAMPLE below, please notice

that the heading line of the MONITORING REPORT (FAST SVC) does say INACTIVE which means this service had been deactivated by the operator prior to the issuance of the SIR MON command and that, as a result, the counters are not maintained any more.

Note: Before you start (another) monitoring cycle, please make sure that you have issued the **SYSDEF TD,RESETCNT** command **while monitoring was active** (MON=ON). Only if monitoring is active it will be ensured that all the counters are reset to zero.

```
sir mon
AR 0015                      MONITORING REPORT (SVC)
AR 0015                   (based on a 0013:44:35.960 interval)
AR 0015 EXCP     =    3645  FCH-$$B  =        178  SVC-03   =          9
AR 0015 LOAD     =    1389  FCH-$$A  =          8  WAIT     =       5843
AR 0015 SVC-08   =       4  LBRET    =          4  SETIME   =          1
AR 0015 SVC-0B   =      74  SVC-0C   =         44  SVC-0D   =       1079
AR 0015 EOJ      =       6  SYSIO    =       2894  STXIT IT =          1
AR 0015 EXIT IT  =       1  STXIT OC =          1  SVC-16   =        218
AR 0015 SETIME   =    9878  SVC-1A   =         27  WAITM    =      19914
AR 0015 COMREG   =    2217  GETIME   =        185  FREE     =          9
AR 0015 STXIT AB =       2  ATTACH   =          2  DETACH   =          2
AR 0015 POST     =       1  DYNCLASS =         15  SVC-2C   =          1
AR 0015 HIPROG   =       4  TTIMER   =          9  GETPRTY  =          8
AR 0015 INVPART  =      12  GETVIS   =       2390  FREEVIS  =       2350
AR 0015 CDLOAD   =     122  RUNMODE  =          3  PFIX     =         11
AR 0015 PFREE    =       1  REALAD   =         22  SETPFA   =          1
AR 0015 SECTVAL  =     543  SVC-4C   =          2  ALLOCATE =         12
AR 0015 SETLIMIT =      44  SVC-5B   =          4  XECBTAB  =         11
AR 0015 EXTRACT  =      49  GETVCE   =        738  MODVCE   =         51
AR 0015 EXTENT   =      33  SUBSID   =         11  FASTSVC  =      34177
AR 0015 SECHECK  =     133  (UN)LOCK =       3205  MSAT     =        267
AR 0015 XPCC     =      30  VIO      =         15  NPGR     =          2
AR 0015 PGSER    =       2  SYSDEF   =          1  PRODID   =        199
AR 0015 SVC-83   =     129  SVC-85   =          4  VSIUCV   =          9
AR 0015 INACTIVE             MONITORING REPORT (FAST-SVC)
AR 0015  FC-02   =     121     FC-06 =         49     FC-1B =         48
AR 0015  FC-1E   =      63     FC-31 =          1     FC-3F =         55
AR 0015  FC-40   =      41     FC-42 =          5     FC-44 =          9
AR 0015  FC-46   =      55     FC-47 =         56     FC-49 =          5
AR 0015  FC-4A   =       8     FC-4B =          8     FC-4E =         17
AR 0015  FC-4F   =      10     FC-51 =         35     FC-59 =          2
AR 0015  FC-5D   =       1     FC-5F =          1     FC-62 =          4
AR 0015  FC-67   =       8     FC-73 =         60     FC-74 =          6
AR 0015  FC-7D   =       2     FC-7E =          1     FC-85 =         12
AR 0015  FC-90   =      98     FC-96 =          1     FC-AA =          1
AR 0015            CPU 0000   PERFORMANCE COUNTERS
AR 0015   CNT-00 =  105746    CNT-01 =     110743    CNT-02 =     110743
AR 0015   CNT-03 =   84683    CNT-08 =      18544    CNT-09 =       4999
AR 0015   CNT-0B =   26629    CNT-0D =      34851    CNT-0E =     112566
AR 0015                       MONITORING REPORT (BOUND STATE)
AR 0015   BND-49 =       3    BND-50 =      12429    BND-80 =        220
AR 0015   BND-82 =     440    BND-85 =        179    BND-87 =          1
AR 0015 1I40I  READY
```

Figure 35. SIR MON command sample

SIR MIH

The SIR MIH command provides the capability to enable, disable the MIH (Missing Interrupt Handler) process in z/VSE. In addition the command may be used to adjust the cycle time after which the MIH process in z/VSE is activated or to show the MIH settings. The MIH process in VM, if running under VM, remains UNAFFECTED by this command.

The MIH cycle time may be set for single device or in general. There are two variants of SIR MIH command to be used dependent on which type of MIH cycle times (single device or general), you want to manage.

Syntax(general): **SIR MIH[,{nnnnnn|ON|OFF}]**

MIH indicates that MIH settings are to be displayed or altered.

- `nnnnnn` is the numeric value in seconds, after which the MIH process is to be (re-)initiated. This is called the MIH cycle time which is normally set to 180 seconds. Any value between 1 and 999999 will be accepted.
- `ON` enables MIH processing.
- `OFF` disables MIH processing. The MIH process is to be totally bypassed, and as a result no more lost interrupts, if any, will be recognized.

SIR MIH without any additional parameter displays the current effective general MIH cycle time settings.

Syntax(single device): **SIR MIH [,cuu[{=nn[M]|,{ON|OFF}}]**

MIH indicates that MIH settings are to be displayed or altered.

- `cuu` is the device number for which MIH setting is to be altered/displayed.

  Note: MIH command processing on a single device base requires an explicit MIH cycle time setting for this device before the OFF/ON option is valid.
- `nn` is the numeric value in seconds or minutes, after which the MIH process is to be (re-)initiated.The number may be appended by 'M' for minutes. Any value between 1 and 59 will be accepted.
- `ON` indicates that the MIH process with the existing constants is to be re-activated.
- `OFF` indicates that the MIH process is to be totally bypassed, and as a result no more lost interrupts, if any, will be recognized.

SIR MIH,cuu without any additional parameter displays the device specific MIH cycle time settings in effect.

The following sample should document the functional capabilities, but it does NOT mean that you have to follow the given sample in order to change the MIH setting. You can of course change the MIH setting regardless of whether MIH is OFF or ON, but the new setting will become active only if MIH is ON.

The command will produce similar output as shown in the sample below.

```
sir mih
AR 0015 MIH ON    INTERVAL=    5  SECONDS
sir mih=off
AR 0015 MIH PROCESSINsir mih=offVE
sir mih
AR 0015 MIH OFF   INTERVAL=    5  SECONDS
sir mih=180
AR 0015 MIH TIME INTERVAL SET TO 180    SECONDS
sir mih
AR 0015 MIH OFF   INTERVAL=  180  SECONDS
sir mih,on
AR 0015 MIH PROCESSING NOW ACTIVE
sir mih
AR 0015 MIH ON    INTERVAL=  180  SECONDS
sir mih,261
AR 0015 1I04I INVALID COMMAND
sir mih,261=40
AR 0015 1I40I  READY
sir mih,261
AR 0015 MIH ON    INTERVAL=    40 SECONDS
sir mih,641=5m
AR 0015 1I40I  READY
sir mih,641
AR 0015 MIH ON    INTERVAL=  300 SECONDS
```

Figure 36. SIR MIH command sample

SIR CHPID

Syntax: **SIR CHPID**

This command provides information about the Channel Pathes (CHPID) defined to your system. This command may cause the Attention Routine to be canceled in case your processor or even your VM Release does NOT SUPPORT that request, like the PC Server 500 System/390.

The command will produce similar output as shown in the sample below.

```
sir chpid
AR 0015 CHPID CHLA SWLA LSN      CHANNEL-PATH-DESCRIPTION
AR 0015  00                  PARALLEL BYTE-MULTIPLEXER
AR 0015  01                  PARALLEL BLOCK-MULTIPLEXER
AR 0015  02                  PARALLEL BLOCK-MULTIPLEXER
AR 0015  03                  PARALLEL BLOCK-MULTIPLEXER
AR 0015  04                  PARALLEL BLOCK-MULTIPLEXER
AR 0015  08                  PARALLEL BLOCK-MULTIPLEXER
AR 0015  18   01             SERIAL POINT-TO-POINT
AR 0015  19                  FIBER EXTENDED CHANNEL
AR 0015  18   01             SERIAL POINT-TO-POINT
AR 0015  19                  FIBER EXTENDED CHANNEL
AR 0015  1A             01   FIBER EXTENDED CHANNEL
AR 0015  1B                  SERIAL CHANNEL PATH (INIT)
AR 0015 CHPID CHLA SWLA LSN      CHANNEL-PATH-DESCRIPTION
AR 0015  1C   C4   FE   01   SERIAL SWITCHED POINT-TO-POINT
AR 0015  1D   C5   FE   01   SERIAL SWITCHED POINT-TO-POINT
AR 0015  1E   C6   FE   01   SERIAL SWITCHED POINT-TO-POINT
AR 0015  1F   01             SERIAL POINT-TO-POINT
AR 0015 1I40I  READY
```

Figure 37. SIR CHPID command sample

Explanation:

CHPID is the CHannel Path ID to which this information applies.

CHLA CHannel Link Address indicates the link address assigned to the channel-subsystem link-level facility of the specified channel path (CHPID). This is used by control units as the destination link address when attempting to communicate with the channel subsystem on the specified channel path.

SWLA SWitch Link Address contains the link address assigned to the control unit link-level facility of the dynamic-switch. This is used as the destination link address when attempting to communicate with the dynamic-switch control unit on the specified channel path (CHPID).

LSN Logical Switch Number gives the logical switch number of the switch, if any, that is attached through the corresponding Channel-Path Link Address (CHLA).

SIR CRWMSG

Syntax: **SIR CRWMSG={ON|OFF}**

This command activates CRW (Channel Report Word) information to be provided on the system operator console. A CRW will typically be presented in case devices are being ATTACHed or DETACHed. The processing of CRW's itself remains unaffected by this command.

CRWMSG is the keyword required to display Channel Report Word information.

- **ON** activates the facility to get CRW Information displayed on the system operator console.
- **OFF** no further CRW information will be provided on the system operator console.

The command will produce similar output as shown in the sample below.

```
sir crwmsg=on
AR 0015 CHANNEL REPORT MESSAGES are ENABLED

#cp def 480 481
TAPE 0481 DEFINED

AR 0017 CRW  TYPE  REPS  ERCC  REPSID   DVNUM
AR 0017       00    03    84    0010    0481

sir crwmsg=off
AR 0015 CHANNEL REPORT MESSAGES are DISABLED
```

Figure 38. SIR CRWMSG=ON command sample

Explanation:

TYPE Basically contains information about the nature of this CRW; if it is a solicited, an unsolicited or a chaining CRW and also identifies if an CRW overflow had occurred.

REPS Is the REPorting-Source identification Code. This could have been the monitoring facility (02), the subchannel (03 as in our sample), the channel path (04), the configuration alert facility (09) or the subchannel itself (0B).

ERCC Is the Error Recovery Code if, when zero, indicates that the channel subsystem has presented event information, otherwise this code identifies the recovery state of the reporting-source code (REPS). In our sample (84)

it indicates that the installed parameters for the given subchannel (ref: REPSID) have been initialized.

REPSID Is the REPorting-Source-ID which, together with the Reporting-Source-Code, either contains the affected subchannel number or it contains the Channel-Path-Id. In the sample,(0010) it is the subchannel that had been initialized.

DVNUM Is the Device-Id of the device as seen by the z/VSE system (cuu address) that is affected by this CRW.

SIR PMRMON

Syntax: **SIR PMRMON={ON|OFF}**

This command activates page manager monitoring counters to be provided on the system operator console.

PMRMON is the keyword required to display the page manager monitoring counters.

- **ON** resets the page manager monitoring counters and activates the monitoring facility.
- **OFF** deactivates the page manager monitoring facility.

The command will produce output as shown in the sample below.

```
sir pmrmon=on
AR 0015 PAGE MANAGER MONITORING HAS BEEN RESET AND ACTIVATED
AR 0015 1I40I  READY

sir pmrmon
AR 0015                 PAGE MANAGER MONITORING REPORT
AR 0015              (BASED ON A 0000:00:04.348 INTERVAL)
AR 0015 IPFQ 31-BIT     =          0   IPFQ 64-BIT     =          0
AR 0015  PSQ 31-BIT     =      10656    PSQ 64-BIT     =          0
AR 0015 PF EXCH TOTAL   =          2   PF EXCH 31->64  =          0
AR 0015 PF EXCH 64->31  =          0   PGFLT TOTAL     =       3340
AR 0015 PGFLT PMGR      =       3336   PGFLT USER      =          0
AR 0015 PGFLT IMM PO 31 =        376   PGFLT IMM PO 64 =          0
AR 0015 SELCT ON PSQ 31 =       1670   SELCT ON PSQ 64 =          0
AR 0015 SELC R=1 MAX 31 =         66   SELC R=1 MAX 64 =          0
AR 0015 RECLAIMS        =         59   NPSQ LOW        =          0
AR 0015 PGOUT I/O TOTAL =        955   PGIN I/O TOTAL  =          0
AR 0015 PGOUT I/O UNC.  =        376   PGOUT I/O PRE.  =        579
AR 0015 LRA PGM CHECK   =          0   TFIX 64-BIT FR  =          0
AR 0015 HWM MB FRM-64   =          0   HWM MB FRM-31   =        950
AR 0015 1I40I  READY

sir pmrmon=off
AR 0015 PAGE MANAGER MONITORING HAS BEEN DEACTIVATED
AR 0015 1I40I  READY
```

Figure 39. SIR PMRMON command sample

SIR VMCF

Syntax: **SIR VMCF[={ON|OFF}]**

This command allows to display the status of the VMCF interface and to alter it. Use the command to turn the VMCF interface on after it has been turned off inadvertently e.g. by a CMS user using VSECMD with incorrect authorization. Message VMCF12I CMS-Z/VSE CONSOLE INTERFACE DEACTIVATED will inform the operator that the interface has been turned off.
SIR VMCF=ON will restart the interface.

STACK

Syntax: **STACK(P)**

The STACK command is a command that can be used for different purposes.

- It is a command that enables the system operator to prepare a sequence of commands and/or replies, give it a name and have this sequence executed whenever that name is being entered or submitted as a command.
- It is a command that enables the system operator to suppress or change any z/VSE command.
- It is a command that enables the system operator to abbreviate long z/VSE commands to just a few characters.

The STACK command will cause the associated data to be saved temporarily (just in storage) and as a result the data is lost after the next subsequent IPL. If, however, the STACKP command (STACK Permanently) had been used instead, any data that needs to be saved will be written onto the CKD device that had been chosen for the last IPL and as a result will be maintained across further IPL processes. The STACKP will not work on FBA devices from which z/VSE may have been IPLed.

The initial idea was to provide a means to shut-down a z/VSE system by using a single command as for example:

STACK SHUT|MSG F2|23 CEMT P,SHUT|Z NET QUICK|PEND|1 ROD
SHUT would be the name of the command that the operator would have issued to close down CICS, VTAM and finally POWER.

I would suggest that you start your STACK exploration on a test system first, of course not before you have read the description given on the next few pages.

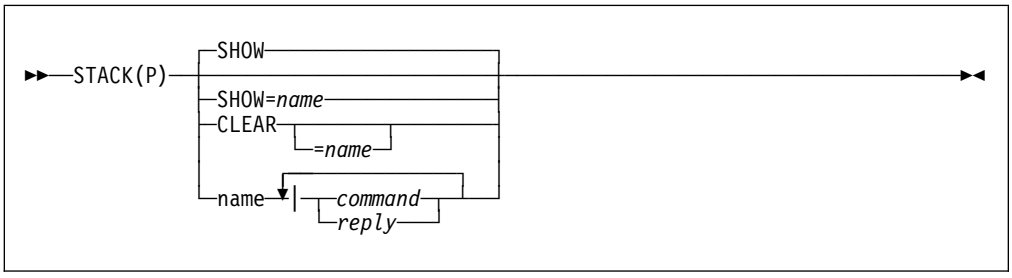


Figure 40. STACK command syntax

STACK identifies this command as a STACK command.

name This is a STACK-name that the user can freely choose and that he would have to specify to get the sequence of the specified commands and/or replies executed in the specified order. The whole entry is referred to as a STACK-entry and it will be saved in a special STACK-buffer for subsequent use and the whole STACK-buffer will be preserved throughout IPL. The preserved STACK data can only be reset or purged via the STACK CLEAR command. It should be noted that a STACK-name does supersede a z/VSE command with the same name until this STACK command has been removed (see option CLEAR).

CLEAR This is a special keyword indicating that STACK entries currently defined in the STACK-buffer are to be cleared. If the user does not supply a STACK-name, then all STACK-entries (the whole STACK-buffer), will be purged, whereas, only the named STACK entry will be purged otherwise. The option CLEAR is required to enforce the system to also purge any eventually preserved STACK-data (a single STACK-entry versus the total STACK-buffer).

SHOW This is a special keyword indicating that STACK entries currently defined in the STACK-buffer are to be displayed on the operator console. If the user does not supply a STACK-name, then all STACK entries (the whole STACK-buffer), will be displayed, whereas, only the named STACK entry will be displayed otherwise. SHOW is the default option that will be executed in case the STACK command has been issued without any further operands.

Note: Caution when you use the STACK command without any further operands and you are not sure of whether you have applied the fix for APAR DY45629. It could cause your STACK-buffer to be purged unexpectedly.

command is a complete z/VSE-command which may have incorporated special variables (&0 through &9) which will be explained later. The command must be followed by a vertical bar (|) which serves as a separator.

reply is a complete z/VSE-reply which may contain incorporated special variables (&0 through &9) which will be explained later. The reply must be followed by a vertical bar (|) which serves as a separator. A system control statement is considered a reply in this sense.

You can concatenate as many commands or replies assuming the whole STACK command does not exceed 126 bytes, which is the max. z/VSE command buffer length. None of the operands in the STACK command will be validated at the time it is being entered, it will be saved unchanged into the STACK-buffer. An eventually existing STACK-entry with an identical STACK-name will be overwritten unconditionally. You may specify up to a max. of 32 STACK-entries. Any of the STACK-entries may contain variables, identified by &n where n is an integer ranging from 0 through 9. These variables will eventually be substituted by operands at execution time, which is when the STACK-name is being entered as a command. &0 will get the first positional parameter assigned, &1 the second positional parameter and so on. If a parameter is being omitted at execution time, then the appropriate variable will be omitted also.

To get a better understanding lets start with a simple command.

STACK VIS|GETVIS &0,ALL

This command will cause z/VSE to save a STACK-entry with the name **VIS** in the STACK-buffer and will subsequently cause z/VSE to execute a **GETVIS id,ALL** whenever **VIS id** is being entered on the operator console. The following samples will show how the &0 variable will be replaced by the parameter submitted in the **VIS** command.

```
stack vis|getvis &0,all
AR 0015 1I40I  READY

vis sva
    AR 0015 1I40I  READY
    AR 0015 GETVIS SVA,ALL
    AR 0015 GETVIS USAGE   SVA-24    SVA-ANY                      SVA-24    SVA-ANY
    AR 0015 AREA SIZE:    1,404K     4,880K
    AR 0015 USED AREA:      272K       608K MAX. EVER USED:        312K       648K
    AR 0015 FREE AREA:    1,132K     4,272K LARGEST FREE:        1,112K     3,140K
    AR 0015 SUMMARY REPORT
    AR 0015 SUBPOOL       REQUEST  <--SVA-24-AREA---    --SVA-ANY-AREA-->
    AR 0015 Default                              128K                   4K
    AR 0015 IPWPWR                               32K                   0K
    AR 0015 IJBFCB                               16K                   0K
    AR 0015 INLSLD                               12K                   0K
    AR 0015 IJBPRC0020    SPACE                   8K                   0K
    •
    •
    •
    AR 0015 IJBCSC                                0K                   4K
    AR 0015 VMCFSP                                0K                   4K
    AR 0015 IJBCSM                                0K                 136K
    AR 0015 ILSTCK                                0K                   4K
    AR 0015 SUBPOOL TOTALS                      272K                 288K

    vis f1
    AR 0015 1I40I  READY
    AR 0015 GETVIS F1,ALL
    AR 0015 GETVIS USAGE    F1-24     F1-ANY                       F1-24     F1-ANY
    AR 0015 AREA SIZE:      832K       832K
    AR 0015 USED AREA:      140K       140K MAX. EVER USED:        192K       192K
    AR 0015 FREE AREA:      692K       692K LARGEST FREE:          684K       684K
    AR 0015 SUMMARY REPORT
    AR 0015 SUBPOOL       REQUEST  <---F1-24-AREA---    ---F1-ANY-AREA-->
    AR 0015 Default                              120K                   0K
    AR 0015 IPWGEN                                 8K 0K
    AR 0015 SUBPOOL TOTALS                       128K                   0K
    AR 0015 1I40I  READY
```

Figure 41. STACK command sample 1

Now lets add another STACK-entry to prevent e.g. REIPL commands to be executed as valid z/VSE commands. We could do this by issuing a:
STACK REIPL|* REIPL command has been received.
Any REIPL command would thus be made a simple comment statement. The simple fact that we named the STACK-entry **REIPL** did it. As mentioned earlier, STACK-names supersede z/VSE command names and as a result the REIPL command will be transformed into a comment line as the following sample shows.

```
stack reipl|* reipl command has been received
AR 0015 1I40I  READY
           reipl 120,noprompt
           AR 0015 1I40I  READY
           AR 0015 * REIPL COMMAND HAS BEEN RECEIVED
```

Figure 42. STACK command sample 2

As the above sample shows, the initial **REIPL 120,NOPROMPT** command has been transformed into the comment statement as we defined it in the STACK-command.

If we now issue the STACK SHOW command, we would get both the entries that we have issued so far.

```
stack show AR 0015 VIS|GETVIS &0,ALL
AR 0015 REIPL|* REIPL COMMAND HAS BEEN RECEIVED
AR 0015 1I40I  READY
```

If we just want to SHOW the STACK-entry with the name REIPL, we would have to submit the following command.

```
stack show=reipl
AR 0015 REIPL|* REIPL COMMAND HAS BEEN RECEIVED
AR 0015 1I40I  READY
```

Now lets add a third sample where we want to abbreviate a z/VSE command to just a few characters. As a sample I have used the POWER command to release a PAUSE Job. As you see the variable &0 has been appended to the PAUSE-Job, which means it will be substituted at execution time as the different samples will show.

```
STACK R|PRELEASE RDR,PAUSE&0
AR 0015 1I40I  READY


    r bg
    AR 0015 1I40I  READY
    AR 0015 PRELEASE RDR,PAUSEBG
    AR 0015 1C39I COMMAND PASSED TO VSE/POWER
    F1 0001 1R88I  OK
    BG 0001 1Q47I BG PAUSEBG 00235 FROM (SYSA), TIME=13:54:10, TKN=00000295
    BG 0000 // JOB PAUSEBG
            DATE 10/20/1997, CLOCK 13/54/10
    BG-0000 // PAUSE

    r f2
    AR 0015 1I40I  READY
    AR 0015 PRELEASE RDR,PAUSEF2
    AR 0015 1C39I COMMAND PASSED TO VSE/POWER
    F1 0001 1R88I  OK
    F2 0001 1Q47I F2 PAUSEF2 00027 FROM (SYSA), TIME=14:00:55, TKN=00000296
    F2 0002 // JOB PAUSEF2
            DATE 10/20/1997, CLOCK 14/00/55
    F2-0002 // PAUSE

    r
    AR 0015 1I40I  READY
    AR 0015 PRELEASE RDR,PAUSE
    AR 0015 1C39I COMMAND PASSED TO VSE/POWER
    F1 0001 1R88I  NOTHING TO RELEASE
```

Note: No job was released because no such job-name (PAUSE) did exist.

Figure 43. STACK command sample 3-1

If we had used the following STACK-command, then this command would **NOT HAVE WORKED** and we would have received an error message because of the fact that **R** would have called **R** again and recursion is not allowed.

```
STACK R|R RDR,PAUSE&0
AR 0015 1I40I  READY

R BG
AR 0015 1I40I  READY
AR 0015 R RDR,PAUSEBG
AR 0015 1I40I  READY
AR 0015 R RDR,PAUSEBG
AR 0015 1P44I PREVIOUS 'STACK   '  COMMAND IGNORED
AR 0015 1I40I  READY
```

Figure 44. STACK command sample 3-2

To get this erroneous STACK-entry deleted, we issue the

```
stack clear=r
AR 0015 1I40I  READY
```

The last sample is a more complex one. It does use three STACK-entries, named LIBR, LI and LD and will cause a job to be executed automatically whenever LIBR with the appropriate parameter(s) is being entered as command. Just see the three samples below.

```
stack libr|r rdr,pausebg|0 exec libr|0 acc s=ijsysrs.syslib|&0
AR 0015 1I40I  READY
stack ld|0 ld &1.proc|0 end|0 /&|0
AR 0015 1I40I  READY
stack li|0 l &1.proc|0 end|0 /&|0
AR 0015 1I40I  READY
libr li,$asiproc
F1 0001 1R88I  OK
BG 0001 1Q47I BG PAUSEBG 00236 FROM (SYSA), TIME=16:09:08, TKN=00000297
BG 0000 // JOB PAUSEBG
        DATE 10/20/1997, CLOCK 16/09/08
BG-0000 // PAUSE
0 EXEC LIBR
BG 0000 L001A ENTER COMMAND OR END
BG-0000
0 ACC S=IJSYSRS.SYSLIB
BG 0000 L113I RETURN CODE OF ACCESS IS  0
BG 0000 L001A ENTER COMMAND OR END
BG-0000
0 L $ASIPROC.PROC
BG 0000
BG 0000 MEMBER=$ASIPROC.PROC     SUBLIBRARY=IJSYSRS.SYSLIB    DATE: 97-10
BG 0000                                                       TIME: 16:09
BG 0000 ---------------------------------------------------------------------
BG 0000 CPU=FF0880009672,IPL=$IPL088,JCL=$$JCL088   /* ESA
BG 0000 CPU=FF0890009672,IPL=$IPL089,JCL=$$JCL089   /* ESA
BG 0000 CPU=FF08A0009672,IPL=$IPL08A,JCL=$$JCL08A   /* ESA
BG 0000 CPU=FF08B0009672,IPL=$IPL08B,JCL=$$JCL08B   /* ESA
BG 0000 CPU=FF0010009672,IPL=$IPLESA,JCL=$$JCL001   /* ESA
BG 0000 CPU=FF08C0009672,IPL=$IPLESA,JCL=$$JCL08C   /* ESA
BG 0000 CPU=FF18C0009672,IPL=$IPLESA,JCL=$$JCL18C   /* ESA
BG 0000 CPU=FF18C0009672,IPL=$IPLESA,JCL=$$JCL18C   /* ESA
BG 0000 L113I RETURN CODE OF LIST IS  0
BG 0000 L001A ENTER COMMAND OR END
0 END
BG-0000
BG 0000 1S55I  LAST RETURN CODE WAS 0000
BG-0000 1I00D  READY FOR COMMUNICATIONS.
0 /&
BG 0000 EOJ PAUSEBG   MAX.RETURN CODE=0000
        DATE 10/20/1997, CLOCK 16/32/36, DURATION   00/00/11
BG-0000
0
AR 0015 1I40I  READY
AR 0015 R RDR,PAUSEBG
AR 0015 1C39I COMMAND PASSED TO VSE/POWER
AR 0015 LI
AR 0015 1I40I  READY
BG 0000 EOJ NO NAME
        DATE 10/20/1997, CLOCK 16/32/36
BG 0001 1Q34I   BG WAITING FOR WORK
```

Figure 45 (Part 1 of 3). STACK command sample 3-3

```
libr ld,$asiproc
F1 0001 1R88I  OK
BG 0001 1Q47I BG PAUSEBG 00252 FROM (SYSA), TIME=18:57:30, TKN=00000298
BG 0000 // JOB PAUSEBG
        DATE 10/20/1997, CLOCK 18/57/30
BG-0000 // PAUSE
0 EXEC LIBR
BG 0000 L001A ENTER COMMAND OR END
BG-0000
0 ACC S=IJSYSRS.SYSLIB
BG 0000 L113I RETURN CODE OF ACCESS IS  0
BG 0000 L001A ENTER COMMAND OR END
BG-0000
0 LD $ASIPROC.PROC
BG 0000
BG 0000 DIRECTORY DISPLAY        SUBLIBRARY=IJSYSRS.SYSLIB    DATE: 97-10
BG 0000                                                       TIME: 18:57
BG 0000 ------------------------------------------------------------------
BG 0000  M E M B E R      CREATION   LAST     BYTES    LIBR CONT SVA  A-
BG 0000 NAME     TYPE     DATE      UPDATE   RECORDS   BLKS STOR ELIG MOD
BG 0000 ------------------------------------------------------------------
BG 0000 $ASIPROC PROC     97-05-28 97-05-28     19 R       2 YES   -   -
BG 0000 L113I RETURN CODE OF LISTDIR IS  0
BG 0000 L001A ENTER COMMAND OR END
BG-0000
0 END
BG 0000 1S55I  LAST RETURN CODE WAS 0000
BG-0000 1I00D  READY FOR COMMUNICATIONS.
0 /&
BG 0000 EOJ PAUSEBG   MAX.RETURN CODE=0000
        DATE 10/20/1997, CLOCK 18/57/31, DURATION   00/00/00
0
AR 0015 1I40I  READY
AR 0015 R RDR,PAUSEBG
AR 0015 1C39I COMMAND PASSED TO VSE/POWER
AR 0015 LD
AR 0015 1I40I  READY
BG 0000 EOJ NO NAME
        DATE 10/20/1997, CLOCK 19/02/29
BG 0001 1Q34I   BG WAITING FOR WORK
```

Figure 45 (Part 2 of 3). STACK command sample 3-3

```
libr ld,* F1 0001 1R88I  OK
BG 0001 1Q47I BG PAUSEBG 00254 FROM (SYSA), TIME=19:06:25, TKN=00000299
BG 0000 // JOB PAUSEBG
        DATE 10/20/1997, CLOCK 19/06/25
BG-0000 // PAUSE
0 EXEC LIBR
BG 0000 L001A ENTER COMMAND OR END
BG-0000
0 ACC S=IJSYSRS.SYSLIB
BG 0000 L113I RETURN CODE OF ACCESS IS  0
BG 0000 L001A ENTER COMMAND OR END
BG-0000
0 LD *.PROC
BG 0000
BG 0000 DIRECTORY DISPLAY        SUBLIBRARY=IJSYSRS.SYSLIB    DATE: 97-10
BG 0000                                                       TIME: 19:06
BG 0000 ------------------------------------------------------------------
BG 0000  M E M B E R      CREATION   LAST     BYTES    LIBR CONT SVA  A-
BG 0000 NAME     TYPE     DATE      UPDATE   RECORDS   BLKS STOR ELIG MOD
BG 0000 $8JCL001 PROC     97-09-23 97-09-23     10 R       1 YES   -   -
BG 0000 $9JCLESA PROC     97-05-28   -  -       10 R       1 YES   -   -
BG 0000 $9JCL001 PROC     97-09-23 97-09-23     10 R       1 YES   -   -
BG 0000 ALLOC    PROC     97-05-28 97-09-23     27 R       1 YES   -   -
BG 0000 ALLOCESA PROC     97-05-28 97-09-23     14 R       1 YES   -   -
BG 0000 CPUVAR1  PROC     97-05-28   -  -       62 R       2 YES   -   -
BG 0000 DLI1A0   PROC     97-09-23 97-09-23     71 R       4 YES   -   -
BG 0000 DLZSAMJS PROC     97-09-23 97-09-23      6 R       1 YES   -   -
•
•
•
BG 0000 STDLABEL PROC     97-05-28 97-05-28     66 R       4 YES   -   -
BG 0000 STDLABUP PROC     97-05-28   -  -      124 R       6 YES   -   -
BG 0000 STDLABUS PROC     97-05-28   -  -       17 R       1 YES   -   -
BG 0000 STDLB001 PROC     97-09-23 97-09-23     71 R       4  NO   -   -
BG 0000 STDPROF  PROC     97-05-28   -  -       11 R       1 YES   -   -
BG 0000 USERBG   PROC     97-05-28   -  -       22 R       1 YES   -   -
BG 0000 L113I RETURN CODE OF LISTDIR IS  0
BG 0000 L001A ENTER COMMAND OR END
BG-0000
0 END
BG 0000 1S55I  LAST RETURN CODE WAS 0000
BG-0000 1I00D  READY FOR COMMUNICATIONS.
0 /&
BG 0000 EOJ NO NAME
        DATE 10/20/1997, CLOCK 19/06/32
BG-0000
0
AR 0015 1I40I  READY
AR 0015 R RDR,PAUSEBG
AR 0015 1C39I COMMAND PASSED TO VSE/POWER
AR 0015 LD
AR 0015 1I40I  READY
BG 0000 EOJ NO NAME
        DATE 10/20/1997, CLOCK 19/06/32
BG 0001 1Q34I   BG WAITING FOR WORK
```

Figure 45 (Part 3 of 3). STACK command sample 3-3

Notes:

- If for some reason you should ever have problems with the STACK processing, then you just need to issue a **STACK CLEAR** command to get the STACK processing deactivated.
- In case of problems you can issue a **CANCEL AR** command to get back control.
- While STACK processing is ongoing, you can not normally enter other z/VSE commands. To do so, you have to issue the **RC** command which will cause the currently ongoing STACK command processing to be suspended.
- If you are running a z/VSE-system with fix-level DY45926 or above, then the STACK-buffer will be preserved throughout subsequent IPL's.
- The whole STACK-buffer information will be lost in case you have NOT applied the fix for APAR DY45629 and your system has been newly IPLed. In this case, to ensure that you have always your STACK-entries set properly, you could run a simple job at start-up-time which could set up the STACK-entries by using the SVC-30 interface.

  I have appended such a sample program on the next page. Once it has been assembled and catalogued, you can execute it during your system startup-procedure and it will load the STACK entries similar to what we discussed on the previous pages. The only purpose of this sample is to give you an idea of what can be done and how it should be done.
- You need of course to make modifications to this program (mainly the STACK1 and subsequent constants) to make it suit your installations needs. You can of course also use this skeleton program to submit any other z/VSE command during system-start-up time.
- The vertical bar '|' has the hexadecimal value 4F in EBCDIC.

```
         TITLE 'VSE/ESA     STACK SET PROGRAM'
         PUNCH ' PHASE STACKSET,*  '
         SPACE 1
         START X'78'
TESTBEG  BALR  RD,0
         USING TESTBEG+2,RD       BASE REGISTER
         B     TEST35A0           SKIP CONSTANTS
         DC    C'SAMPLE TO SET STACK ENTRIES (AXEL PIEPER 1997)'
         DC    CL8'STACKSET'
         DC    CL8'INITIAL '      CHANGE ACTIVITY
         SPACE 2
TEST35A0 SLR   R0,R0              SET FUNCTION CODE
         LA    R1,SV30PARM        POINT TO PARAMETER LIST
         LA    RE,(STACK1-SV30PARM)/6 LOOP COUNT
TEST35B0 SVC   30                 SUBMIT THE COMMAND
         CH    RF,H4              WAS TASK BUSY
         BE    TEST35B0           YES, TRY (UNLIMITED)
         LA    R1,6(,R1)          ADVANCE TO NEXT ENTRY
         BCT   RE,TEST35B0
         EOJ                      END OF PROGRAM
H4       DC    H'4'               CONSTANT
SV30PARM DC    AL2(STACK2-STACK1),AL4(STACK1)
         DC    AL2(STACK3-STACK2),AL4(STACK2)
         DC    AL2(STACK4-STACK3),AL4(STACK3)
         DC    AL2(STACK5-STACK4),AL4(STACK4)
         DC    AL2(STACK6-STACK5),AL4(STACK5)
         DC    AL2(STACKN-STACK6),AL4(STACK6)
* APPEND SIMILAR ENTRIES
* UP TO A MAXIMUM OF 32
         EJECT
STACK1   DC    C'STACK VIS|GETVIS &&,ALL'
STACK2   DC    C'STACK REIPL|* REIPL command has been received'
STACK3   DC    C'STACK R|PRELEASE RDR,PAUSE&&0'
STACK4   DC    C'STACK LIBR|R RDR,PAUSEBG|0 EXEC LIBR|'
         DC    C'0 ACC S=IJSYSRS.SYSLIB|&&0'
STACK5   DC    C'STACK LD|0 LD &&1.PROC|0 END|0 /&&|0'
STACK6   DC    C'STACK LI|0 L &&1.PROC|0 END|0 /&&|0'
* APPEND SIMILAR ENTRIES
* UP TO A MAXIMUM OF 32
STACKN   DS    0H'0'
         EJECT
R0       EQU   0
R1       EQU   1
R2       EQU   2
RC       EQU   12
RD       EQU   13
RE       EQU   14
RF       EQU   15
         END
```

Figure 46. STACK SET program

STATUS

Syntax: **STATUS cuu[,ALL]**

The STATUS cuu command allows retrieval of device status information and, if appended with the ALL option, will also provide addressing information about the related I/O control blocks. This command is especially useful in case of device hang conditions.

The command will produce similar output as shown in the sample below.

```
status 480,all
  (1.0) SCHIB DEV  INT-PARM ISC FLG LP PNO LPU PI MBI  PO PA CHPID0-3
  (2.0) 0010  0480 00003710   3  83 80  00  80 80 0000 80 80 0AFFFFFF
  (3.0)            KEY SLCC FPIAUZEN FCTL ACTL SCTL CCW-ADDR DS CS CNT
  (4.0)             0   0      00     40  0000  07  00027358 0C 00 000C
  (5.0) REQUEST IS STARTED             DEVICE IS BUSY
  (6.0)   PUB=00003710   PUBX=0007A228   PUB2=00072288   POWN=00003BBC
  (6.0)  VCTE=000712FA  POWNX=00237BC4
AR 0015 1I40I  READY
```

Figure 47. STATUS command sample

Explanation

(1.0) this line is the heading line describing the information provided in the next line.

- SCHIB this field contains the Subchannel Number of the device that has been inspected.
- DEV this field contains the device number (cuu) which is being used within z/VSE to identify and address the device.
- INT-PARM this field contains the INTerrupt-PARaMeter that had been passed in the ORB at SSCH time.
- ISC this field contains the I/O Interruption Subclass Code that has been assigned to this subchannel. z/VSE normally only uses the subclass code three, except for SDAID, which has a different subclass code assigned.
- FLG this field is the FLaG byte which identifies the status plus some features that have been enabled for this subchannel.
- LP this field identifies the Logical-Path(es) that z/VSE did allow the channel subsystem to use in accessing the device.
- PNO this field identifies the Path(es) that the channel subsystem found Not Operational when attempting to address the device via the identified path(es).
- LPU this field identifies the Last-Path-Used by the channel subsystem to communicate with the device.
- PI this field identifies the Path(es)-Installed for this device (as defined in the IOCDS).
- MBI this field is the Measurement-Block-Index used by the channel subsystem to calculate the address of the measurement block for this subchannel, assuming that measuring is active (see SIR SMF).
- PO this field identifies the Path(es) that where found Operational by the channel subsystem last time it did use that path.
- PA this field identifies the Path(es) that where found Available but not necessarily operational by the channel subsystem.
- CHPID0-3 and CHPID4-7 these fields contains all the channel path Id's that have been defined (IOCDS) to access a certain device.

(2.0) this line is telling us that device X'480' with the subchannel number x'0010' and the interrupt parameter X'00003710' has been ENABLED (FLG=8x) for I/O interrupts for subclass 3 and can only be accessed via a single path (PI=X'80') with the CHannel-Path-ID X'0A'.

(3.0) this line is the heading line of dynamic subchannel information provided in the next subsequent line. This line as well as the next line will only be presented if an I/O interrupt is outstanding, or if they had explicitly been requested (option ALL or DEBUG active).

- KEY — defines the storage protection KEY that had been used at SSCH time.
- SLCC — this field contains information about the progress of the I/O operation. The important information is the cc-bits, which is the deferred condition code of the I/O operation in progress if it is unequal to zero.
- FPIAUZEN — this field contains control bits which are of minor interest.
- FCTL — this field contains Function ConTroL information which is:
  - 40 start function
  - 20 halt function
  - 10 clear function
- ACTL — this field contains Activity ConTroL information where the first byte contains PENDING I/O instruction information
  - 04 start pending
  - 02 halt pending
  - 01 clear pending

  and the second byte contains the subchannel/device activity information
  - 80 subchannel active
  - 40 device active
- SCTL — this field contains Status ConTroL information where the bits have the following meaning assigned.
  - 10 ALERT status which normally indicates an error condition or the presentation of an unsolicited interrupt.
  - 04 primary I/O interrupt status which normally indicates the completion of data transfer operations.
  - 02 secondary I/O interrupt status which indicates the completion of an I/O operation at the device level.
  - 01 status pending which indicates that I/O interrupt status as defined by the other SCTL bits is PENDING in the subchannel and waiting to be presented to the z/VSE system as soon as interrupts are enabled.
- CCW-ADDR — this field contains the address+8 of the last CCW that had been executed.
- DS — this field contains the device status information
- CS — this field contains the channel status information
- CNT — this field contains the residual count, that is the number of bytes that had not been transferred.

(4.0) this line is now telling us, that a SSCH operation (FCTL=40) has just completed its operation since primary plus secondary I/O interrupt status is pending (SCTL=07), which means it has not yet been presented to the system. The last executed CCW was at address X'27350 and has completed normally (CE+DE interrupt). 12 bytes of data (CNT=000C) had not been transferred.

(5.0) this line is simply repeating in plain text form the status of the device at the time it was interrogated. It should be self explanatory. REQUEST IS STARTED DEVICE IS BUSY; this is in sync with what we found out before

(ref:4.0), because the system has not yet received the I/O interrupt so from its point of view, the device is still busy.

(6.0) this line contains the addresses of some I/O control blocks which might be interesting to know.

TIME

NOTE: It is recommended to use the commands SET and SET ZONEDEF/SET ZONEBDY as described in Systems and Control Statements publication and to IPL the system to change the system time. The TIME command should only be used by experts who know all the specific dependencies of system time changes made on the fly for all running software. Please checkout the results of the TIME command in a test system environment first before using it in a production system.

Syntax: **TIME**
[DATE=mm/dd/yyyy,CLOCK=hh/mm/ss][ZONE={EAST|WEST}/hh/mm]

The TIME command allows you to display or alter the current setting of the Time-Of-Day (TOD) clock and/or the time ZONE. It is a functional equivalent to the SET DATE.... command as described in the System Control Statements Manual.

However, special precautions must be adhered to, when using the TIME command in conjunction with other operand(s) because changes to the TOD or ZONE could impact subsystems or vendor programs products running under z/VSE and could also impact Job Accounting. For that purpose it is recommended to prevent TOD or ZONE updates **WHILE** such subsystems are up or while Jobs are running. Results could be unpredictable. See also “Effect of Day-Light Saving Time Changes” on page 84.

TIME
[DATE=mm/dd/yyyy,CLOCK=hh/mm/ss][ZONE={EAST|WEST|zid}/hh/mm]

Note: zid this operand exist for VSE/ESA 2.3.x only.

The **TIME ZONE=VM** command has been added as a valid command and it will cause the currently active VM-ZONE settings to be immediately applied as the new z/VSE system zone. Eventually active partition ZONE settings will be adjusted according to the difference between the OLD z/VSE system zone setting and the NEW system zone setting (as retrieved from VM).

```
time
  (1.0) TIME IS: 08:41:56 (GMT + 2  H) DATE 10/08/1997  WEDNESDAY
time date=10/26/97,clock=11/11/00
  (2.0) TIME/ZONE HAS BEEN UPDATED
  (3.0) TIME IS: 11:11:00 (GMT)        DATE 10/26/1997  SUNDAY
time zone=CET
  (2.0) TIME/ZONE HAS BEEN UPDATED
  (4.0) TIME IS: 12:11:09 (CET)        DATE 10/26/1997  SUNDAY
```

Figure 48. TIME command sample

Explanation:

(1.0) this line displays the current LOCAL time as the result of the immediately preceeding time command.
(2.0) this line indicates that the time or zone has been updated.
(3.0) this line displays the LOCAL time as the result of the immediately preceeding time command. Note that the ZONE has been forced to GMT (Greenwich Mean Time) since no explicit ZONE operand was given.
(4.0) this line displays the LOCAL time as the result of the immediately preceeding time command. Note that the ZONE has been altered to CET (Central European Time) which must have been defined to the z/VSE system via the ZONEDEF operand during IPL.

Effect of Day-Light Saving Time Changes

Many countries operate a policy of adjusting clocks by one hour at the beginning and end of Summer to effect what is commonly referred to as 'day-light saving'. These time changes are also applied to the local times held in computers. Generally, all hardware clocks are set to Greenwich Mean Time (GMT). An offset value representing the local time-ZONE is maintained separately by the Operating Systems (z/VSE) and is required to properly calculate the LOCAL TIME. It is this offset value that is adjusted when making day-light saving time changes, while the hardware clock remains unchanged.

Adjusting Local Time Forwards

A local time change forwards has normally no effect on subsystems operations, nor should it cause problems in IBM supplied, or user written accounting routines.

For example, moving the clock forward an hour has no effect on CICS partitions, or the way CICS reads back through the system log during an emergency restart.

Adjusting Local Time Backwards

A local time change backwards could affect subsystems or user written job accounting routines more severely.

CICS

A local time change backwards can affect the operation of a CICS emergency restart in the event of an abnormal termination that occurs after the time change. If the system log at the time of the CICS failure contains active units-of-work that span the point at which local time was changed, CICS emergency restart will fail to recover the relevant data for transaction backout owing to the discontinuity in time-stamps in the system log.

A local time change could also affect the operation of forward recovery utility programs on forward recovery logs since all CICS system log and journal records are time stamped in local time. For a CICS emergency restart to work correctly, the time stamps of the active records on the system log must not go backwards in time-that is, the records must be in chronological order. If you adjust local time backwards by an hour while CICS is running, the time stamps on system log and journal records become out of sequence. You can avoid problems with emergency restarts during a time change by adopting the following procedure:

- Shut down all CICS partitions, either normally or forced with the IMMEDIATE option, before changing local time.

- Adjust the local time backwards.
- Wait for the length of time by which the local time was adjusted.
- Restart the CICS regions.

VOLUME TAPE

Syntax: **VOLUME TAPE**

A new flavor of the already existing VOLUME command has been provided, which is the VOLUME TAPE commands. This command does provide information about the TAPE devices that had been ADDed (or sensed) during IPL processing. The information that the VOLUME TAPE command provides slightly differs from the normal VOLUME command output because it contains additional information that should be useful for system operators.

For more information refer to the *System Control Statements Manual*.

The command will produce similar output as shown in the sample below.

```
volume tape
 (1.0)  CUU  CODE DEV.-TYP   VOLID  USAGE   SHARED    STATUS  CAPACITY
 (2.0)  480  5400  3490-40   TAP634 BG                  BUFD     22356
 (2.5)                              COMPR.-RATIO =   3.83
 (3.0)  481  5400  3490-40   *NONE* F4                  SYNC         0
 (4.0)  482  5408  3490-40   ISMINE BG              2XF SYNC         1
 (5.0)  483  5400  3490-40          UNUSED          NOT READY
 (6.0)  484  5400  3490-40          SHARED
```

Figure 49. VOLUME TAPE command sample

Explanation:

`CUU` specifies the device number under which the device is known to the z/VSE system.

`CODE` contains the z/VSE device type code and the mode setting which is currently active for this tape device.

`DEVICE-ID` is the device-type and model information of the tape device.

`VOLID` is the VOL1 label (if any) of the media currently or last mounted on the tape drive. If no VOL1 label exists, *NONE* will be displayed.

`USE/STATUS` in difference to the normal VOLUME command, this field contains information about the owner of the tape device if an owner exists. This is normally the partition Id (BG,FG1,F2...,xn).
UNUSED indicates that there is NO current user.
SHARED indicates that the device is currently being OWNED by another processor or LPAR.

`INFORMATION` in difference to the normal VOLUME command, this field contains information about the media format mode 2XF versus XF and whether the media and the CU are currently operating in SYNC or in BUFD (synchronous versus buffered) mode. BUFD would indicate that record(s) are still in the CU buffer. This information is applicable to the 3480/3490(E) and 3590 only.

`CAPACITY` in difference to the normal VOLUME command, this field contains information about the last block (record) which has been read from, or written to the storage media. This information is applicable to the

3480/3490(E) and 3590 only and could be used to examine the usage and progress of the tape operations. A value of zero would indicate that the media is positioned at Load-Point (LP).

(1.0) CUU CODE DEVICE-ID VOLID USE/STATUS-INFORMATION CAPACITY
This is the standard heading line for any VOLUME command.

(2.0) 480 5400 3490-40 TAP634 BG BUFD 22356
This line indicates that tape drive X'480', which is a 3490-40 type device with the VOL1 label of TAP634 is owned by partition BG and is running with mode X'00' (Data Compaction OFF). It has processed 22356 records and buffered data is available in the CU buffer.

(2.5) COMPR.-RATIO = 3.83
This line is optional and only present for 3480/3490 devices with IDRC. It indicates that the data that had been written to the cartridge has been compacted by the factor nn.nn (e.g. 3.83) whenever a value smaller than 1.00 is encountered, it would be an indication that the overhead, generated and appended to the user-data is greater as the savings due to compaction. This would typically be the case for small block (record) sizes. If the compaction is 1.00, that would be the indication that your records have been written uncompacted. Any value greater 1.00 would be real savings due to compaction. For 3590 type devices, there will be an additional field appended to the compaction ratio and this field will provide information on how much of the tape cartridge has already been occupied. Repeatedly issued VOLUME TAPE commands will get you more accurate data as your output file grows.

(3.0) 481 5400 3490-40 *NONE* F4 SYNC 0
This line indicates that tape drive X'481', which is a 3490-40 type device without a VOL1 label is owned by partition F4 and is running with mode X'00' (Data Compaction OFF). It has processed zero (0) records and is thus positioned at Load-Point.

(4.0) 482 5408 3490-40 ISMINE BG 2XF SYNC 1
This line indicates that tape drive X'482', which is a 3490-40 type device with the VOL1 label of ISMINE is owned by partition BG and is running with mode X'08' (Data Compaction). It has processed a single record (positioned behind the VOL1 label) and the CU does not have any buffered data (SYNC).

(5.0) 483 5400 3490-40 UNUSED NOT READY
This line indicates that tape drive X'483', which is a 3490-40 type device is currently UNUSED and that the device is NOT READY. The currently mode setting is Data-Compaction OFF.

(6.0) 484 5400 3490-40 SHARED
This line indicates that tape drive X'484', which is a 3490-40 type device is currently ASSIGNED ELSEWHERE and can thus not be accessed by this processor/LPAR.

Chapter 3. z/VSE Turbo Dispatcher and Its Exploitation

If you already know how the Turbo Dispatcher works, you may start reading with “Turbo Dispatcher Considerations” on page 91.

Turbo Dispatcher Design

The Turbo Dispatcher can utilize multiprocessors by distributing the workload across several processors (CPUs) of one Central Electronic Complex (CEC), enabling them to work in parallel and thus increase the overall throughput of a z/VSE system.

Note: Since VSE/ESA 2.4 the Turbo Dispatcher is the only supported dispatcher. Before VSE/ESA 2.4 the standard dispatcher could be selected.

The z/VSE Turbo Dispatcher works on a partition (job) basis, that is, it dispatches an entire partition to a CPU waiting for work, instead of dispatching at a subtask level like z/OS does. Subsequently "jobs" is used as a synonym for partitions. One job consists of many work units. A **work unit** is defined as a set of instructions that are executed from the selection by the z/VSE Turbo Dispatcher until the next interrupt. Only one work unit of a job can be processed at a time, that is, no other work unit of the same job can run on a different CPU. This means for jobs with multitasking applications (applications with attached z/VSE subtasks), that no other task of the same job can execute on a different CPU, when one task of that job is already active.

There are two different work unit types:

1. **parallel work units (P)**
   Most customer applications in batch as well as the online (CICS) environment are processed as parallel work units. However, when an application calls a supervisor service, it has to process a non-parallel work unit in most cases.
2. **non-parallel work units (N)**
   Most system services and key 0 applications (such as supervisor- and ACF/VTAM services) will be processed as non-parallel work units.

   Only one CPU within the CEC may process a non-parallel work unit at a time, that is as long as this work unit type is active, no other CPU can execute a non-parallel work unit. Any other job, that wants to process a non-parallel work unit, has to wait until no other CPU processes a non-parallel work unit.

   However, other CPUs may process parallel work units of other jobs.

   Notes:

   a. Work units of the VSE/POWER maintask can be processed in parallel, if the VSE/POWER autostart statement

      ```
      SET WORKUNIT=PA
      ```

      is specified in the VSE/POWER startup procedure.

   b. Some vendors adapted their applications to run parallel work units even if they are executing in key zero.

The following simplified example (no interrupts are considered, all work units have the same length) should give you an impression, how the z/VSE Turbo Dispatcher processes a given workload (see Figure 50 on page 88). Job A, B and C are

ready for selection, job A has highest, job C lowest priority. Each job consists of 3 work units, e.g. job A consists of work units A1, A2, and A3. Work unit A1 has to be processed before work unit A2 (of the same job) can be selected. Work units are either parallel (P) or non-parallel (N). On a uni-processor the three jobs would need 9 process steps (3 jobs times 3 work units), the z/VSE Turbo Dispatcher would need only *5 steps* on a 2-way (dyadic) CEC as shown in the example (with CPU 0 and CPU 1):

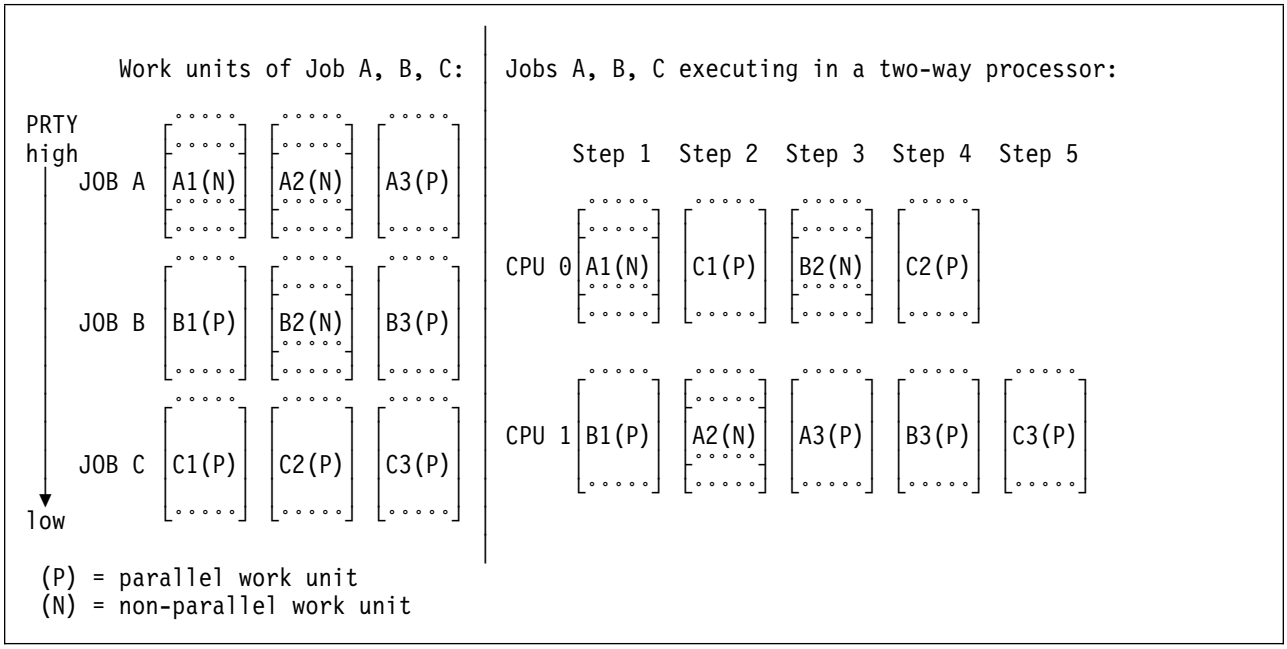


Figure 50. Processing Steps for Jobs A, B and C

- *Step 1:*
  CPU 0 selects non-parallel work unit A1, at the same time CPU 1 selects parallel work unit B1, because job A and B have a higher priority than job C.
- *Step 2:*
  The next two highest priority work units are A2 and B2. However, both work units are non-parallel and therefore cannot run at the same time. So the next lower priority parallel work unit C1 will be selected, that is this step is made of work unit A2 and C1. A2 may be selected by CPU 0 or CPU 1, in the example it executes on CPU 1.
- *Step 3:*
  Now non-parallel work unit B2 and parallel work unit A3 will be processed, as given by their priority.
- *Step 4:*
  Job A terminated. C2 and B3 are both parallel work units and will be processed by CPU 0 and CPU 1.
- *Step 5:*
  Now also job B terminated. In our example CPU 1 processes the last available work unit C3 of job C.

Advantages on Uni-Processors

The Turbo Dispatcher provides advantages not only for multiprocessors but also for uni-processors in terms of

- multiprocessor exploitation prediction,
- measurement tools (SIR command),
- partition balancing enhancements,
- performance improvements for some environments (e.g. when VSE/ICCF is active) and
- preparation for future z/VSE releases, where the standard dispatcher is no longer available.

Partition Balancing Enhancements

The Turbo Dispatcher provides an improved partition balancing algorithm, which gives each partition, be it static or dynamic, equal weight within a balanced group. The PRTY command defines the balanced group, e.g.

```
PRTY F4,C=BG=F5,F3,F2,f1
```

In our example we have dynamic class C (which may hold up to 32 dynamic partitions), static partitions BG and F5 as **balanced group members**.

Balanced group members are time sliced. The calculation of a time slice size is based on the MSECS interval. The MSECS command may change this MSECS interval (default is about 1000 milliseconds).

With the standard dispatcher each balanced group member will receive a time slice. So dynamic partitions of dynamic class C would only get a fraction of the time slice (e.g. when C holds 3 dynamic partitions, only a third of the time slice). When the time slice expires, the corresponding balanced group member will receive a lower priority. That is, when the time slice of the dynamic class expires, the whole class (including all corresponding dynamic partitions) will receive a lower priority.

With the Turbo Dispatcher active all dynamic and static partitions of the balanced group will receive the same time slice. That is when a dynamic partition's time slice expires, only this partition will be moved to a lower priority, all other dynamic partitions of the same class will not change their priority.

Relative CPU Shares

The PRTY SHARE command allows to specify a **relative share** of CPU time for each static partition and dynamic class belonging to the balanced group. With this enhancement it is much easier, for example, to balance a CICS partition with batch partitions in a way that ensures acceptable throughput for the batch partitions and acceptable response times for the CICS/TS transactions.

Quiesce CPUs

This Quiesce support is especially implemented for z/VSE systems running as guests under z/VM to reduce the overhead of stopped CPUs. It is possible to quiesce a CPU, that is a CPU will be suspended from task selection, but from a z/VM view the CPU is still active. A quiesced CPU which is not needed during a certain period of time (for example during off-shift) helps to minimize the overhead caused by idle additional CPUs.

CPU Balancing (z/VSE 4.2)

CPU Balancing may reduce the multiprocessing overhead, because the Turbo Dispatcher only selects CPUs, that are required for the current workload. CPU Balancing may also reduce the overhead or performance degradation for I/O intensive workloads running in just one partition (other partitions are idle) with multiple CPUs active.

With new SYSDEF parameters you can specify the interval, after which active/quiesced/inactive CPUs may be activated/quiesced/stopped, and a percentage value (threshold) for the CPU utilization.

z/VSE 4.2 will provide two options for CPU balancing

1. CPU balancing via quiesce (STOPQ)

   This option is implemented based on the available "Quiesce CPU" process. With the SYSDEF TD, STOPQ= command you may quiesce a CPU. That is all CPUs that are quiesced will no longer receive work units, I/O interrupts are disabled. CPU balancing will automatically quiesce CPUs dependent on the overall CPU utilization and a given time interval, if the CPU utilization falls below a given threshold. CPU balancing via quiesce is recommended for LPAR environments and most z/VM guests.

2. CPU balancing via stop (STOP)

   CPU balancing via stop uses the SYSDEF TD start/stop process to manage the available CPUs dependent on the CPU utilization of a given workload. That is if a workload has a lower utilization than the threshold value, only required CPUs to run the workload will stay active other CPUs will be stopped. CPUs will be restarted again, if the workload needs more CPU cycles than the threshold value. Any CPU known to z/VSE independent if the CPU is active, inactive or quiesced - will be eligible for balancing. CPU balancing via stop may be useful in z/VM 5.4 (or higher) environments, which allows a better exploitation of the z/VM guest shares.

The QUERY TD command shows the actual settings.

Example for CPU Balancing:

Assume the interval is 10 seconds, the threshold value 50 (=50 %) and 3 CPUs are active. After the 10 second interval the Turbo Dispatcher verifies, if CPUs need to be added or quiesced.

As long as the overall CPU utilization is below 50 %, just one CPU is active the other 2 CPUs are quiesced. If the CPU utilization increases to 50% or higher, an additional CPU will be activated. If the overall CPU utilization increases to more than 99 %, the 3rd CPU will be activated. If the overall CPU utilization decreases

below 100 %, the 3rd CPU will be quiesed again. Now assume the threshold value is 70. In that case a 2nd CPU will be activated, if the overall CPU utilization increases above 69 %; the 3rd one will be activated above 139 %.

Turbo Dispatcher Considerations

The Turbo Dispatcher support is transparent to most programs as well as IBM's subsystems such as CICS and ACF/VTAM. Apart from few exceptions, application programs can run functionally unchanged with the Turbo Dispatcher. However, there may be the need to adapt applications for better multiprocessor exploitation (e.g. by implementing larger I/O buffers or using data spaces).

A few system applications run in key zero, have interfaces with the dispatcher or supervisor areas or update the first 4 KB page. These applications may have the need for adaptations. Examples for such applications are performance monitors, accounting and scheduler routines.

Note: The traditional replacement of the **SVC new PSW** (Program-Status Word) e.g. by vendors cause performance degradations in the multiprocessor environment. z/VSE provides vendor exits to get rid of that replacement.

Vendor products adapted their applications to the Turbo Dispatcher environment and improved performance (compared to the standard dispatcher in former VSE/ESA releases) by exploiting vendor exits.

Most user applications are written in high level languages (such as COBOL) and do not access internal system areas.

Before you migrate to a multiprocessor environment you should consider your hardware and software requirements:

- Does my largest partition still fit into a single CPU of the target processor ?
- Is the processor capacity and speed still sufficient to run the workload ?
- Does multiprocessing help to run the work load ?
- Is there a need to remove an I/O bottleneck or to add devices ?
- Do my vendor products run on or exploit the Turbo Dispatcher ?
- Do I have system applications that interface with system routines or areas ?
- Determine your non-parallel share (NPS) via the QUERY TD command.
- Try to exploit the ESA functions like 31 bit addressing for larger I/O buffers or data spaces to **reduce the non-parallel share** via less I/Os, which will improve your multiprocessor exploitation.
- What is my expectation level ?

Note: Whenever possible you should not change hardware and software in one step.

How to Calculate the Processor Size and the Number of CPUs

In general, workload migration from one release to the next release will increase the overall CPU time by a few percent, which is also dependent on the exploitation of new functions. In VSE/ESA Version 2 the Turbo Dispatcher on a uni-processor will have 5 to 12 percent overhead compared to the standard dispatcher running the same workload. The overhead is caused by the multiprocessing implementation and depends on the workload, e.g. an I/O intensive workload has more overhead as a CPU intensive workload. Our measurements showed a Turbo Dispatcher overhead of 12 % for a very I/O intensive batch work load and 5% for an online (CICS) workload. The Turbo Dispatcher with multiple CPUs active will increase the CPU time, part of that increase is caused by the spin time (SPIN_TIME, see Figure 52 on page 93). The spin time should be low compared to the overall CPU time.

The following figure shows migration steps into the n-way environment, that may impact CPU time:

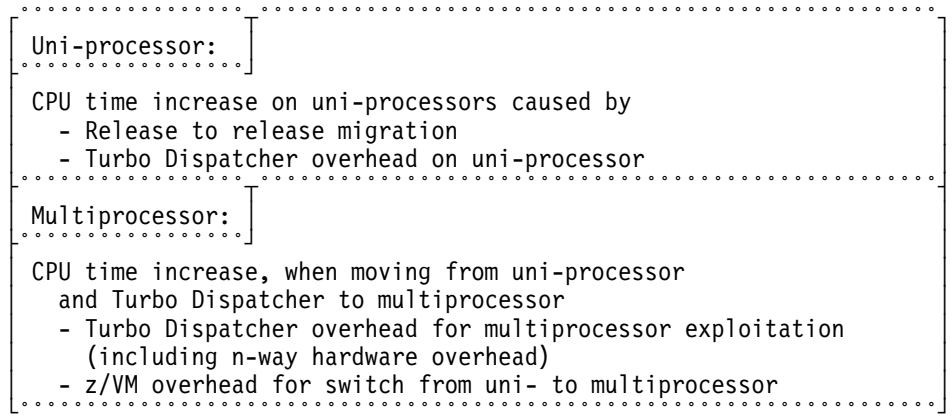


Figure 51. Migration Steps that will Impact CPU Time

Note: If you run your system with **DEBUG ON**, you may need 10 to 30 % more CPU time.

If your capacity planning shows that multiprocessing with the Turbo Dispatcher could be the right choice to run your workload, the following considerations can help in getting the right processor size:

- Determine the maximum requirements (e.g. response time, number of concurrent partitions/users, resource consumption of largest partition, etc.) for your current VSE system. Consider the additional overhead as described and the expected growth.
- Use your "largest" VSE partition (e.g. CICS) to determine the minimum size of one CPU, because one partition can only run on one CPU at a time. For that purpose you may get the required data out of performance monitors or (job) accounting information.
- Determine the maximum number of CPUs that can be exploited by your workload. You may use the QUERY TD command to get the non-parallel share of your workload or any performance monitor. A QUERY TD output example for an I/O intensive workload is shown below:

```
CPU   STATUS   SPIN_TIME  NP_TIME    TOTAL_TIME    NP/TOT
 00   ACTIVE      5137     1466775       3350836     0.437
 01   ACTIVE       559     1443508       3648302     0.395
 02   QUIESCED       0           0             0     *.***
 03   INACTIVE
                ------------------------------------------
TOTAL             5696     2910283       6999138     0.415

                NP/TOT: 0.415      SPIN/(SPIN+TOT): 0.000
 OVERALL UTILIZATION: 154%          NP UTILIZATION: 63%

 ELAPSED TIME SINCE LAST RESET:  4548364


The CPU time values are shown in milli seconds for all partitions:

  TOTAL_TIME - total CPU time
  NP_TIME    - non-parallel CPU time, contained in TOTAL_TIME
  SPIN_TIME  - spin CPU time waiting for a resource occupied
               by another CPU, not contained in TOTAL_TIME
  NP/TOT     - non-parallel share = NP_TIME / TOTAL_TIME
  SPIN/(SPIN+TOT) = spin time ratio
```

Figure 52. QUERY TD Output Example

Note: Replacement of the **SVC new PSW** (Program-Status Word) will increase the SPIN_TIME.

I/O intensive workloads have a higher non-parallel share as CPU intensive ones.
The non-parallel share of a given workload is closely related to the amount of key 0 code to be processed and is an indicator for the number of CPUs that can be exploited at a maximum, if enough partitions are available:

Non-parallel Workload Share NP/TOT	0.30	0.35	0.40	0.50	0.55
Max. No. of Fully Exploitable CPUs	3.0	2.6	2.2	1.8	1.6

Figure 53. Relationship of Non-Parallel Workload and Exploitable CPUs

The actual number of CPUs exploited depends on the system resources needed by your workload and is usually less than the maximum values shown in table Figure 53.

Our experience of the past years showed that todays customer workloads can exploit up to 3 CPUs (dependent on NP/TOT).

The non-parallel share of an online (CICS) environment ranges from 0.20 to 0.45 and for a heavy I/O intensive batch workload from 0.45 to 0.55.

So a 2-way system will be a good choice for an I/O intensive environment, where Data-In-Memory (DIM) was not exploited.

- For multiprocessing exploitation there should be enough partitions available to get a high multiprocessor exploitation. So you may consider to add more batch workload (partitions) to your system or additional CICS partitions. If possible, you may also split large CICS partitions into smaller ones. These partitions may be independent or connected via MRO (Multi Region Option).

The following figure shows examples of our measurement results for batch and online workloads:

	Online Workload	Batch Workload
Workload type	3 CICS (2-way) 4 CICS (3-way) with 300 simulated terminals per CICS	8 static + 8 dyn. partitions with e.g. DL/I, SORT, COBOL applications 7 batch jobs/part. very I/O intensive
Non parallel share	0.29	0.45
TD overhead on uni-processor	about 4.5 %	about 12 %
2-way versus uni-processor throughput (CPU utilization)	1.75 (92 %)	1.5
3-way versus uni-processor throughput (CPU utilization)	2.3 (82 %)	---

Figure 54. Examples of our Measurement Results

Note: All performance data contained in this publication were obtained in a specific environment and are presented as an illustration. The results obtained in other operating environments may vary.

More Information

You will find further details about the Turbo Dispatcher on the z/VSE home page: ibm.com/vse *www.ibm.com/systems/z/os/zvse/*

Chapter 4. Console

Console Types

When talking about the z/VSE console then normally one thinks about the z/VSE system console. However, there are more consoles than just the system console where you IPLed from. This is particular important to remember when your system console hung up for whatever reason. Then you still have your other consoles from where you can operate your z/VSE system and eventually make your system console work again.

These are the consoles you usually have:

1. VSE system console. This is the console where you IPLed from.
2. IUI console. There can be many IUI consoles, however, this requires VTAM and CICS to be up and operational.
3. VMCF console. This console is only available when running under VM.
4. Integrated console. An integrated console must be supported by the hardware. This is the case with all newer systems. The integrated console function is available also under VM.
5. Workstation Console If you are using the VSE/ESA workstation feature, you can also have a workstation console. This console requires CICS/APPC for communication.

Please be aware that z/VSE does not support the Integrated 3270 console.

z/VSE System Console

The z/VSE system console is the console where you IPLed from. On VM the z/VSE console screen layout is depending on the terminal definition and can be a line mode or a fullscreen console. A fullscreen console requires *'TERM CON 3270'.* With *'TERM CON 3215'* z/VSE will IPL in console line mode.

If you run natively, the console will be the terminal where you created the first interrupt (by pressing the enter key) after IPLing from the service processor. The native console must be a local non-SNA terminal. A local non-SNA terminal is not absolutely required to IPL z/VSE. If there is none, then the integrated console can be used instead (even under VM).

The console name is SYS and is displayed in the upper right hand corner of the fullscreen console.

Interactive Interface Console

You can have many Interactive Interface (IUI) consoles. Depending on the authority you defined in the user profile such an IUI console can have master or user authority. Master authority is the highest authority and the same as for the z/VSE system console.

User authority is more restrictive and should be given to all normal users that need to have access to the z/VSE console to monitor jobs, for example.

The console name is the user ID of the IUI user.

VMCF Console

When running under VM, you also can have a VMCF console. In order to work with the VMCF console you must have installed the VM-VSE support on the VM userID from where to use the VMCF console function. Use member SKVMVSE in ICCF library 59 to install this support. After doing this you can use 'VSECMD' to send any z/VSE command to the z/VSE console. The command responses are routed back to the userID from where the command was issued.

The console name is VMC.

Integrated Console

On most systems you will have an integrated console available (please be aware that z/VSE does not support the Integrated **3270** Console). During normal operation this console is dormant. As soon as all consoles are inoperational, the integrated console wakes up automatically.

However, the integrated console can also be used while dormant. This is true no matter whether your z/VSE system runs natively or under VM.

When running natively, the integrated console function is a screen or window on your service processor. Depending on the hardware you are using it might show up differently. You can enter any z/VSE command from the integrated console. The command responses will be routed back to the integrated console, but might show up in a different message window.

VM also emulates the integrated console. To use the integrated console via VM, you must enter

```
VI VMSG vse-command
```

in CP mode under the userID where your z/VSE is running. This works with line mode and fullscreen consoles.

The console name is IC.

Console Diagnostics

There are two ways to get information about the condition of the console router.

1. Use the CORCMD to get information about the status of console queues and consoles.
2. If a problem is suspected in the console router then a console trace can be turned on. To further analyze the trace a dump must be taken.

Using the Trace Feature

The trace is normally turned off. In case of a problem the trace can be turned on by issuing the **DEBUG** or **CORCMD** command.

The easiest method to turn on and off the console trace is to enter **DEBUG ON** and **DEBUG OFF**.

Only if a console trace is needed before the attention routine command handler is available, the CORCMD must be used to turn on the trace.

If no trace area was allocated before, then the default size (currently 8 kB) of the trace area is allocated with this command. This size is normally sufficient. When a different size is needed enter, e.g., *CORCMD TRACE=16* which allocates a trace area of 16 kB.

Note: Do not use a trace area size larger than 31K because the current trace analyzer tool has a limitation of a maximum trace area of 32k. A trace area larger than 32K is rarely needed.

Dump Console Router Trace Area

As soon as the event to be traced has occurred, dump the trace area or better the whole console router storage area. The dump can then be analyzed by a trace utility (IBM internally only). To get the dump do one of the following:

1. Take a standalone dump
2. Take a SVA dump (24 and 31 bit)
3. If under VM, this method can be used as well, which produces the smallest amount of dump. Enter **CORCMD ADDR=DUMP**. Use the address returned by this command as start address of the dump command. Dump up to the end of storage. It is important to dump with translation, e.g.

   ```
   d vt1800000-end
   ```

   since the trace analyzer needs the translated part of the dump.

All console router related storage is in the SVA. Normally the 31-bit SVA is sufficient. In some cases also 24-bit SVA space is used by the console router.

Important Note: Since the trace area is used in wrap-around mode, every activity of the console router after the event occurred must be prevented. Keep in mind that every traffic to and from a console or application program causes console router trace entries to be written. In particular do not perform any redisplay or other commands after the event happened and before the dump was taken. All these activities can cause the trace area to wrap and then important trace information might be lost. Normally, it is OK to issue the CORCMD ADDR=DUMP after the event happened. This command does not cause a lot of traffic. Also just turning off the trace and then do some redisplay before dumping the storage might destroy important information since console router queue items might be reused by this activity.

CORCMD

You can use the **CORCMD** to retrieve console router internal information. No blanks are allowed in the expression following the command name CORCMD. For example, CORCMD ADDR= WA is invalid because of the blank before WA.

Dump and Trace Related Commands

These command options help to manage the console router trace facility and are useful when debugging.

CORCMD ADDR=COR
returns the address where the console router phase $IJBCSIO is loaded.

CORCMD ADDR=TA
returns the address of the console router trace area.

CORCMD ADDR=WA
returns the address of the console router work area.

CORCMD ADDR=DUMP
returns an address from where to dump in case of problem. Dump the storage from this address up to the end of storage. It is important to dump with translation, since the trace analyzer needs the translated part of the dump. E.g. (VM), *dump vt15DD800-end*.

This address is not needed if the entire SVA is dumped or a standalone dump is to be taken. See “Dump Console Router Trace Area” on page 97 for details.

CORCMD TRACE
returns the current trace setting (ON or OFF)

CORCMD TRACE=ON
sets the console router trace to on. If no trace area exists, then a trace area with the default size is allocated (8 kB).

CORCMD TRACE=OFF
sets the console router trace to off.

CORCMD TRACE=END
returns the storage used by the trace area to the system.

CORCMD TRACE=n
changes the size of the trace area. **n** is the value in Kbyte.

Status Information

The following commands are useful to check the status of the console router queues and consoles. In the following explanations the terms RI and ML are used.

RI stands for *Routing Item*. A routing item is a console router queue item to be routed to a destination which can be a console, if it is a message, or a command processor, if it is a command, or an application if it is a reply. A routing item contains one line of message if it is a message routing item.

ML stands for *Message Line*. Message lines are optionally appended to a routing item and contain one line of message. Up to 10 message lines can be appended to a routing item.

CORCMD STATUS=QUEUE
returns status info about the console router queues

```
corcmd status=queue
GETVIS for  RI:  Lim=0020 Cur=0000  ML: Lim=0028 Cur=0000
Non-returnable:  RI=0000007F  ML=000001F3  QMGEmpty: TIK=0021 Code=0001
Returnable  RI:  Lim=0064  Hi=0000  Cur=0000
Returnable  ML:  Lim=0064  Hi=0000  Cur=0000
Alert         :  Pct=0032  RI-Base=00000071  RI-Pct=00000038
CRQ: Cur=001B Hi=001B    MRQ: Cur=0001 Hi=0001    DYQ: Cur=0000 Hi=0000
LRQ: Cur=0019 Hi=0019    DHQ: Cur=0000 Hi=0000
ARQ: Cur=0000 Hi=0001    HCQ: Cur=0000 Hi=0000
FRQ: Cur=0037 Hi=0044    YRQ: Cur=000A Hi=000A    XRQ: Cur=0022 Hi=0031
DOQ: Cur=0002 Hi=0002    XTQ: Cur=0000 Hi=0000    MOQ: Cur=000B Hi=0000
MLQ: Cur=000E Hi=0048    YMQ: Cur=000A Hi=000A    XMQ: Cur=0168 Hi=01A1
End of STATUS=QUEUE
```

Explanation:

```
Lim      max number (e.g. GETVIS)
Cur      current number
Hi       highest number ever (can exceed LIM for returnable RI and ML)
ALERT:    Pct      alert percentage (hex)
          RI-Base  number of RIs to calculate percentage
          RI-Pct   number of RIs used to signal alert
QMGEmpty: TIK      task ID of last unsuccessful request
          Code     QMGempty code of last unsuccessful request:
      00  no free RI or ML found
      01  RI requ for APNormal: too many undeliver. msgs for task (RI
      02  QMEXMLSP: no GETVIS in IPL stage 2
      03  QMEXMLSP: GETVIS failed
      04  QMEXRISP: no GETVIS in IPL stage 2
      05  QMEXRISP: max number of expansions reached
      06  QMEXRISP: GETVIS failed
      07  QMFREE1: CRQ is empty
      08  QMFREE1: cannot be reused since not logged yet
      09  QMFREE1: cannot be reused, blocked by a console
      0A  QMFREE1: no more RIs in CRQ
      0B  QMGETSRI: GETVIS for single RI failed
      0C  QMGETSML: GETVIS for single ML failed
      0D  PROCRIML: Did not get a ML
      0E  PROCRIML: Did not get a ML
      0F  PROCRIML: invalid condition
      10  RI requ for APNormal: too many undeliver. msgs for task (ML
      11  QMFREE1: cannot be reused, blocked by console, RCSUSPND
```

What does it tell:

1. Check this line

   ```
   GETVIS for  RI:  Lim=0020 Cur=0000  ML: Lim=0028 Cur=0000
   ```

   This line tells you whether the console router queues were possibly short on console buffer space. If *Cur=0000* is displayed, then this means that the console router was never short on storage. The first thing the console router does if there is heavy console traffic, is to get more system getvis to expand its buffer storage. Such storage will never be returned to the system. That's why it is limited to the limit shown in *Lim=0020*.

   If *Cur* has reached the value of *Lim* then this tells that the console router buffer were extended to its maximum size. It does not necessarily mean that the console router is currently short on storage, it only tells that it happened at least once since last IPL.

2. Check this line

   ```
   Returnable  RI:  Lim=0064  Hi=0000  Cur=0000
   ```

   When the console router buffer space has extended to its maximum (via GETVIS requests) as shown above, the console router is very

restrictive when it needs more queue items but cannot create it from the current buffer space. In this situation only privileged programs like POWER, CICS, and VTAM will get a queue item to write messages. Also, if a reply is entered at a console the console router will provide a queue item that the reply can be processed.

Such queue items might then be obtained from system getvis one by one as needed. As soon as they are no longer needed they will be returned to the system.

The numbers of *Hi* and *Cur* tell how often the console router needed to create such privileged queue items. If the numbers are 0, then the console router probably was never really in a critical short on storage situation.

The numbers of *Hi* and *Cur* might even exceed the number of *Lim* if required. However, if *Lim* is reached the console router becomes even more restrictive.

3. Check this line

```
CRQ: Cur=001B Hi=001B    MRQ: Cur=0001 Hi=0001    DYQ: Cur=0000 Hi=0000
```

The numbers displayed in *CRQ* show the number of routing items currently in the console router main queue. If this number becomes very low, e.g. less than 10, then there might be a problem.

MRQ shows the number of outstanding replies plus some other special messages (e.g. action messages).

DYQ is the so called delayed message queue. In this queue all routing items are collected that must be kept for a longer period of time. Typically this is the last message of a task, or still outstanding replies, and other special messages like action messages. If this number is very high then this might indicate a potential problem.

4. Check this line

```
FRQ: Cur=0037 Hi=0044    YRQ: Cur=000A Hi=000A    XRQ: Cur=0022 Hi=0031
```

FRQ and *XRQ* show the number of available empty routing items. Any request is first honored from these queues. It is quite normal that these queues are empty after a while. When these queues are empty, the console router tries to reuse the oldest routing item of the queue.

YRQ contains empty routing items for emergency cases. If this queue is empty, then this might indicate a console buffer short on storage problem together with POWER, VTAM or CICS.

5. Check this line

```
Non-returnable:  RI=0000007F  ML=000001F3  QMGEmpty: TIK=0021 Code=0001
```

The *Non-returnable* numbers show the numbers of queue items in non returnable system getvis space.

Check the *QMGEmpty: TIK=0021 Code=0001* values. TIK indicates the task ID of the last requestor for a queue item that was not honored. The reason why it was not honored is shown in *Code*=.

It is quite normal that requests are not honored. In particular *Code=0001* happens frequently. This means that a task tries to write many messages to a console but the console does not retrieve

such messages fast enough. In this case the requesting task is set into a wait condition until the console has retrieved these messages.

Other *Code* values might indicate a problem but don't have to.

It is important to understand that the information shown is the condition of the last unsuccessful request. There is no information available to tell how long ago this happened.

CORCMD STATUS=CONS

returns status info about the consoles known to the console router.

```
corcmd status=cons
Number of consoles:  Act=00000003 Sus=00000000 Ud=00000001 Al=00000001
SYS      001BD3A4 A---      VMC      015DB891 A---      IC       015DB122 A--U
End of STATUS=CONS
```

Explanation:

```
Numer of consoles:  ACTive, SUSpended, accepts UD msgs, accepts ALerts
Status:             Active, Suspended, Netview(automa.), Undel. msgs
```

What does it tell:

This command shows all consoles currently registered to the console router. Typically you will see these consoles.

ConsName	Description
SYS	system console
VMC	VMCF console, only when running under VM
IC	integrated console, only when supported by hardware. All ES/9000 and newer /390 systems support the integrated console.
ConsName	typically the userID of a user, using the console function, or any other name as used by any console application (e.g. vendor products)

CORCMD STATUS=AR

returns status info about the interface between the console router and the AR cmd processor.

```
corcmd status=ar
Stat=Idle  Flag=--  Cons=SYS     -00000001  PIK=0000
ARNxt=N  ARDisp=D  RIPtr=00000000
End of STATUS=AR
```

Explanation:

```
Stat     Idle, Exec, Done, Erro
Flag     X: explanation request; C: cancel request
Cons     console (name & ID) where last cmd was entered
PIK      PIK of console
ARNxt    N: get next from Q; F: get first from Q
ARDisp   Disposition: Keep, Delete, Passed
RIPtr    ptr to last cmd routing item
```

What does it tell: Gives information whether an AR command is currently being executed and how the AR command handler requested the last AR command routing item.

CORCMD STATUS=HCF

returns status info about the interface between the console router and the HCF cmd processor.

```
corcmd status=hcf
Stat= Idle  Stat2: R---1  StatH: HO
PRsn:-L--  CRq:--  CRc:04  HRq:LB  HRc&colon00
End of STATUS=HCF
```

Explanation:

```
Stat     Idle, Exec, Done
Stat2    X---- HCREDCMD (last redisplay cmd)
               R: RED cmd
               E: RED cmd with implicit END
               C: Cancel
               N: Cancel/End
               O: cancel by deactivating console
Stat2    -X--- HCCURREQ (last request from HCF
               H: get next HCF cmd
               L: read log RI
               I: read redisplay RI
               E: get empty RI
               A: add a RI
               U: return a RI
               R: read reply-message
               N: no operation
               C: cleanup - redisplay
               G: cleanup - logging
Stat2    --X-- HCWAIT
               W: HCF set into wait
Stat2    ---X- HCREPLY
               Y: HCF sends reply
Stat2    ----X HC1STLOG
               1: first RI was logged
StatH    H: HCF operational - redisplay possible
         L: logging possible, HC file not open yet
         O: logging possible, HC file open
PRsn     reason why HCF was posted
         A: alert; L: log; C: command; E: empty RI
CRq      last request from console router
         C: cancel; Y: reply to be sent by HCF
CRc      last return code of console router
HRq      last request from HCF cmd processor
         X-: like Stat2 -X---
         -X: last read sub request
               1: read first record for a console
               B: read next record backward
               F: read next record forward
               L: read last (not logged) for a console
               T: validate this record
HRc      last return code of HCF cmd processor
```

What does it tell: Gives information whether a HCF command is currently being executed and how the HCF command handler requested the last HCF command routing item.

CORCMD CONS=name

returns status info about a specific console.

```
corcmd cons=sys
Name=SYS       ID=001BD3A4  Date=1995072  Time=09:14:24.63  Stat=Nrm
RtCd=FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF  MLvl=FE00 Ud=N Al=Y Aut=N DOM=Def
EC=0000  RCS=08000000  CmdPnd=-----  MsgDlv=Srch  CSAFl=---
QFrst=00000000  QSrch=00000000  QDOM=00000000
MsgCt=00000000  DOMCt=00000000  SusCt=00000000
End of STATUS=CONS
```

Explanation:

```
Date & Time of last activate, suspend or resume
Stat     ACTive, SUSpended, RESumed, INItializing, NoRMal
RtCd     enabled routing codes
MLvl     enabled message level
Ud       undeliverable messages accepted: Y or N
Al       alert ECB specified: Y or N
Aut      automatable messages accepted: Y or N
DOM      DOM option: ALL, DEFault, NONe
EC       unique error code for specific return codes
RCS      last return and reason code
CmdPnd   command pending:
         A: AR cmd, H: HCF cmd, C: Console router cmd
         M: HCF cancel/end msg, X: explanation request
MsgDlv   message delivery option: SeaRCH, FIFO, NONe
CSAFl    Console Status Area flags
         A: alerted, U: suspended, X: explain response
QFrst    next message to be delivered
QSrch    search (work) pointer
QDOM     next DOM to be delivered
MsgCt    number of undelivered messages
DOMCt    number of undelivered DOms
SusCt    number of suspends since last activate
```

What does it tell: Besides the information what options a console application used to register to the console router some information can be useful if the status of a console is not clear.

CmdPnd=---- shows whether there is any command pending from this console. If a console does not accept any commands then this might be due to the fact that there is still a previous command pending. A console does only accept a new command if the previous command was entirely processed. The command pending status exists from the time were the command was entered until all of the command response was retrieved by the console.

MsgCt= shows the number of messages that were routed to this console but have not been retrieved yet. The console router expects this console to issue the get_message request to retrieve the outstanding messages

SusCt= shows how often this console was suspended since the last activation of this console. Console suspensions are not unusual. However, if it happens very often, then this indicates that this console does not retrieve messages fast enough. Not retrieving messages can be caused when a console enters the redisplay, help, or explanation mode. If the console is a program (e.g. vendor product) then the get_message request is possibly not issued often enough.

Operational Commands

CORCMD FORCE

CORCMD FORCE checks whether the oldest routing item of the console router main queue (CRQ) is blocked by a console and because of this the reusage of such a routing item is prevented. When this is the case, then such a console is suspended, no matter what console it is. The system responds with:

```
Number of consoles suspended due to CORCMD FORCE : 01
```

CORCMD FORCE=consname

With this command any console can be suspended. This might be useful if a console hung up for any reason. Normally forcing this

console resolves this situation. The forced console can be reactivated by hitting the enter key.

The console name is either the userID of a CICS user or any of the predefined console names (SYS, VMC, IC).

When the system console hung up enter CORCMD FORCE=SYS from any other console.

Special Commands

Be extremely careful with these special commands. Use these commands only if you are exactly knowing what you do or if told so by IBM service personnel.

CORCMD TDALERT=nnnn, CORCMD TDSUSPEND=nnnn

Every console is monitored whether it still receives messages after it was posted that a new message is available. In case a console does not retrieve messages any more the console will be suspended. Before the suspend takes place, the console alert is set. The default time when a console is suspended is about 15 seconds.

Sometimes it can be bothering if console suspension takes place after such a short time, for example while debugging with VM PER. This CORCMD allows to change the time values of the timer driven suspend for alert and suspension.

CORCMD TDALERT=nnnn specifies the time after the console will be alerted.

CORCMD TDALERT=nnnn specifies the time after the console will be suspended.

'nnnn' can be in the range of 5 to 9999. Values below 5 are set to 5. The unit of 'nnnn' is the high order fullword of the clock value which is approx. 1 second.

CORCMD GVLIMRI=nnnn, CORCMD GVLIMML=nnnn

The console router buffer storage management issues system getvis requests if it runs out of buffers. Since the retrieved system getvis will never be returned to the system, the number of getvis requests has been limited. The current limit for routing items is 32 (x'20) getvis requests. Every request is about 4 kB.

With CORCMD GVLIMRI=nnnn the maximum number of getvis request for routing items can be set to a new value.

With CORCMD GVLIMML=nnnn the maximum number of getvis request for message lines can be set to a new value.

'nnnn' can be set to any number between 0 and 9999. Use this command carefully, because the system getvis, once retrieved, will not be returned any more.

Hints and Tips

To better understand some situations you might experience, here is some information about the console support. Most of them have to do with the queueing concept used by the console router. All console traffic among the application programs and consoles goes through the console router queue. When a message enters the console router queue, the target console(s) are posted to retrieve this

message. Only after such a message was retrieved by all target consoles, the queue item occupied by this message is eligible to be reused for a new message.

In general, the oldest queue item is reused for a new message, reply, or command. Therefore, not delivered message queue items are a potential bottleneck in the entire space management of the console router queue. All the following situations have to do with this. To prevent the console router queue from being flooded with messages that cannot be delivered, because a console does not retrieve messages for any reason, two methods are used:

- slow down message writing application programs by setting such a program in a temporary wait condition
- suspend a console that does not retrieve messages

When the Console is flooded by messages for example when a VSE/POWER queue display command has been issued or a LISTDIR for a large VSE Library or a LIST for a VSE library member, you may shorten the process of delivering all messages to the console terminal by entering Redisplay mode. Press PF7 for redisplay which will bypass IO to the console terminal and write the messages to the hardcopy file only.

What Does the Blinking 'MESSAGE' Indicator Mean ?

The 'MESSAGE' indicator shows that there is at least one message available in the console router queue which has not been displayed (retrieved) by this console yet. A reason for not displaying a message can be the console state of redisplay, explanation or help mode a console can enter, or because the user stopped further displaying of messages by setting a console into a HOLD mode.

As the not retrieved message gets older, because new messages are being written, it moves through the queue to finally get reused to build a queue item for another new message. When such an undelivered message passes a certain limit in the queue, the console expected to retrieve this message is alerted. This alert state is indicated by the blinking 'MESSAGE' indicator. Please note that due to the presentation characteristics of some terminals, the blinking state might also be presented by a change in color.

Another reason why the blinking 'MESSAGE' indicator might show up is caused by the response to an AR command. While a master console is in redisplay mode, the user can enter an AR command. Depending on the amount of AR command response messages, the AR command processor might be set into a wait condition because the AR command response messages are not retrieved by the issuing console. Remember that the console where this command was entered, was, and probably still is, in redisplay (or help or explanation) mode and does not retrieve new messages. Since the AR command processor is a valuable resource and no other console can execute a new AR command, this console is alerted as a notification that it might be impacting the whole system.

As a general rule, a console will be alerted after a certain time has elapsed after a message was made available for this console and the console did not retrieve it. The time delay is about 10 seconds. After about 5 more seconds the console will be suspended.

What to do ? If you entered an AR command while not in normal console mode (master console only), be aware that nobody else can execute any AR command

as long as you have not retrieved the entire command response of your AR command by going back into the normal console mode.

The blinking 'MESSAGE' indicator signifies that your console does not retrieve messages available for it because you are not in the normal console mode or your console entered a hold state. The console router might decide to suspend your console soon if necessary. When suspension takes place is not predictable and depends mainly on the amount of messages being written in the system.

What Does Console SUSPENDed Mean ?

Under certain circumstances, the console router can decide to suspend a console if such a console is preventing the reusage of queue items for new messages or input entered at a console. Once it is suspended, such a console is removed from the list of active consoles and all messages destined to this console are considered as delivered.

See also “What Does the Blinking 'MESSAGE' Indicator Mean ?” on page 105 for reasons why a console can be suspended since typically a console will be alerted before it is suspended.

What to do ? A suspended console can resume. If you are using a 3270 console just hit enter. With a CICS console just enter the console dialog again. In any case you have missed those messages that were in the console router queue for this console when suspension took place. This can be any number of messages.

Such messages are not really lost if they were subject to logging. In this case you can use the redisplay function to display the messages you missed.

What Does 'WAITING FOR ROUTER BUFFER SPACE' Mean ?

The console router returns control to an application program after the message was added to the console router queue. This means that a message has not necessarily been displayed at a console screen when the issuing program gets control back. Since consoles can enter a mode where they temporarily do not retrieve messages, for example redisplay mode, programs can continue to write messages to such a console. After a certain number of messages have been queued because they were not retrieved by all target consoles, the issuing program is set into a temporary wait condition. This state is called 'console buffer bound' and you see 'WAITING FOR ROUTER BUFFER SPACE' when entering the STATUS command.

Another situation where you might observe this status is when the reusage of the oldest queue item is prevented by a console that does not retrieve messages and would be a candidate for suspension but cannot be suspended. Such a console can be the system console or any other master console.

What to do ? This state is temporary and disappears as soon as the console(s) that does not retrieve messages resumes retrieval of messages. Master consoles are powerful and are allowed to slow down jobs or the entire system if they do not retrieve messages fast enough.

Therefore, be careful when using multiple master consoles. All master consoles have equal rights, and every user of a master console has the responsibility to let the console work in normal console mode to retrieve all messages.

Do not give master consoles to everybody. Use user consoles instead. A user console has enough authority for most general users.

What is to be Done If a Console Hangs ?

Sometimes a user claims that a console hangs. In this situation, first check whether possibly the whole system is affected (hardwait, loop) or just the console.

If the system is still working, then try to get up a second console. The easiest way normally is to logon to the IUI and use the console function.

Use the **CORCMD FORCE=consname** with the name of the affected console. If the console is the system console the console name is *SYS*. Entering this command normally frees the affected console immediately. Hitting the enter key at the console should reactivate it.

If an IUI console is not available, any other console (See “Console Types” on page 95) can be used. For example you could use one of these consoles:

1. under VM, use the VSECMD to issue the CORCMD.
2. the integrated console, if the hardware supports it. ES/9000 and later systems have an integrated console.
3. VM also emulates the integrated console. Enter *VI VMSG CORCMD ...*
4. Possibly a vendor console product is available.

What Does the Reply-Id Mean ?

Every message that is issued by the system or an application program gets a message prefix assigned in the front of its first line. This message prefix is built in the form

partition_id (+|-) reply_id

where

partition_id takes one of the values AR, BG, Fn, cn (c = dynamic class), and identifies the service owner partition. When the message is issued by a system task or by VSE/POWER, this is the partition that is being serviced (when applicable). In all other cases, it is the partition of the task that issued the message.

+ indicates a messages that needs an immediate reply because a system resource (e.g. the LTA) is being held

- indicates any other message needing a reply

reply_id takes the form **rnnn**, where **nnn** is a 3-digit numerical value derived from the task ID of the task that issued the message (algorithm see below), and **r** is a digit in the range 0 to 9 used to distinguish multiple outstanding replies for the same task. This digit is needed for tasks that are allowed to have more than 1 outstanding reply. Primarily these are subsystems, like CICS or POWER, that use internal tasking.

"nnn"	**Related TaskID**
000 to 011	x'21' (BG Main Task) to x'2C' (FB Main Task)

013	x'01' (Sense Task)
014	x'0B' (ERP Task)
015	x'20' (Attention Routine Task)
016 to 024	x'02' to x'0A' (system tasks)
025 to 044	x'0C' to x'1F' (system tasks)
045 to 512	x'2D' to x'200' (dynamic partitions main tasks or any subtask) You may note that with this group of reply_ids the reply_id value is decimal presentation of the hex task id.

How to stop long-running REDISPLAYs?

Sometimes, when you are searching thru the VSE Hardcopy File (by specifying a filter on the VSE console command line before pressing PF7), it may happen that you already see the expected search results on the screen but searching continues in the background. This may especially happen with large hardcopy files.

In such a situation you may stop searching by using PF6 (CNCL). All console messages which have passed the filter so far are still displayed on the VSE console.

But how to continue then? There are several options:

- Use PF7 and PF8 to scroll thru the filtered console messages
- Or enter the "ALL" filter in the VSE console command line, put the cursor on the redisplay line where you want to start scrolling thru ALL console messages now, and press PF7 or PF8.

Chapter 5. Job Control

Storage Layout and Interaction of Job Control Phases

During end-of-task (SVC14) processing, the terminator routine $IJBSEOT will load the *job control root phase* $JOBCTLA into the partition. The entry point is label JOBCTL, located at X'9000' relative to the partition start address. This code initializes job control and is only executed, when a new copy of $JOBCTLA is loaded into the partition.

When initialization has completed and job control commands are being executed then this initialization code is overlaid by different job control phases $JOBCTLx. Since these other job control phases $JOBCTLx highly depend on the root phase $JOBCTLA, these $JOBCTLx phases are called *job control subphases*.

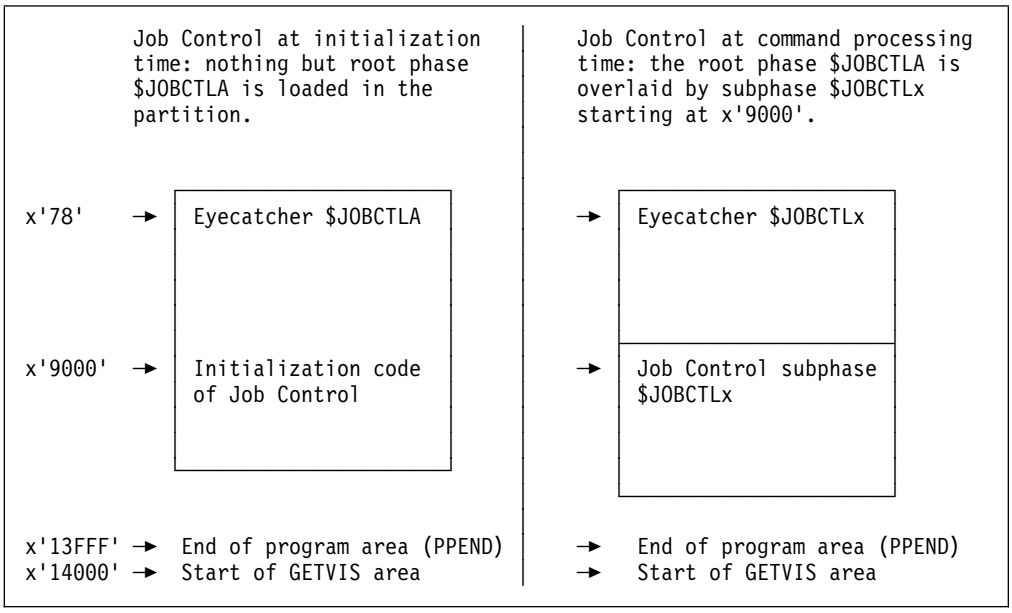


Figure 55. Job Control Storage Layout

Job Control command processing is controlled by $JOBCTLA (subroutine CONTROL). This CONTROL routine

- Reads the next job control command into the command input buffer. This buffer is located at X'1BC' relative to the partition start address.
- Scans the command input buffer for the first operand (i.e. the command name).
- Loops through the command table for a match. This command table contains a 14-byte entry for each available job control command. Each entry contains the letter x identifying the proper job control subphase $JOBCTLx, which will process the command.
- Passes the command to one or more job control user exit routine(s).
- Invokes subroutine EXIT, which will load the command processing subphase at X'9000' relative to the partition start address (provided it is not already in storage) and pass control to the subphase. The subphase will continue to process the command. It will use services provided by the root phase, such as

operand scanning, substitution of symbolic parameters, concatenation of continuation lines, writing of messages to SYSLOG/SYSLST etc. When command processing has completed, the subphase will return to CONTROL.

So each job control command is processed by a subphase.

Figure 56 (Page 1 of 2). Command-to-subphase Cross-Reference

Command/Statement	Corresponding Subphase
ALLOC	$JOBCTLJ
ASSGN	$JOBCTLD
CANCEL	$JOBCTLG
CLOSE	$JOBCTLD
DATE	$JOBCTLJ
DLBL	$JOBCTLK
DVCDN	$JOBCTLF
DVCUP	$JOBCTLF
EXEC	$JOBCTLE
EXTENT	$JOBCTLK
GOTO	$JOBCTLI
HOLD	$JOBCTLJ
ID	$JOBCTLG
IF	$JOBCTLI
IGNORE	$JOBCTLA
JCLEXIT	$JOBCTLO
JOB	$JOBCTLG
KEKL	$IJBCTLO, $IJBKEKL
LIBDEF	$JOBCTLH
LIBDROP	$JOBCTLH
LIBLIST	$JOBCTLH
LIBSERV	$JOBCTLO $IJBSLIB
LISTIO	$JOBCTLF
LOG	$JOBCTLJ
MAP	$JOBCTLO $IJBMAP $IJBMOS
MSECS	$JOBCTLE
MTC	$JOBCTLJ
NOLOG	$JOBCTLJ
NPGR	$JOBCTLJ
ON	$JOBCTLI
OPTION	$JOBCTLG
PAUSE	$JOBCTLA
PROC	$JOBCTLE
PRTY	$JOBCTLO $IJBPRTY
PWR	$JOBCTLE
QUERY	$JOBCTLO $IJBQRY $IJBQUER $IJBMOS
RESET	$JOBCTLF
ROD	$JOBCTLM
RSTRT	$JOBCTLK

Figure 56 (Page 2 of 2). Command-to-subphase Cross-Reference

Command/Statement	Corresponding Subphase
SET	$JOBCTLJ
SETPARM	$JOBCTLE
SETPFIX	$JOBCTLO
SETPRT	$JOBCTLK
SIZE	$JOBCTLJ
START	$JOBCTLG
STDOPT	$JOBCTLJ
STOP	$JOBCTLJ
SYSDEF	$JOBCTLO $IJBSDSP
TLBL	$JOBCTLK
UCS	$JOBCTLJ
UNBATCH	$JOBCTLF
UPSI	$JOBCTLJ
VDISK	$JOBCTLO $IJBVDII
VTAPE	$JOBCTLO $IJBVTAP
ZONE	$JOBCTLJ
*	$JOBCTLA
/.	$JOBCTLA
/+	$JOBCTLG
/&	$JOBCTLG
/*	$JOBCTLA

Notes:

1. The linkage editor commands ACTION, ENTRY, PHASE, MODE, INCLUDE are processed by subphase $JOBCTLJ, that is they are written to SYSLNK.
2. Most Attention Routine commands are allowed if specified in $0JCL.PROC. They are processed by subphase $JOBCTLO, that is they are passed to the attention routine by means of an SVC 30 interface. Symbolic parameters are resolved.
3. 'Modern' job control commands (such as VTAPE, LIBSERV, MAP, PRTY, KEKL, QUERY, SYSDEF, VDISK) are handled by SVA-resident command-processors (such as $IJBVTAP, $IJBSLIB, $IJBMAP, $IJBPRTY, $IJBKEKL, $IJBQRY, $IJBQUER, $IJBSDSP, $IJBVDII, $IJBMOS). The interface between the SVA phase and root phase $JOBCTLA is provided by subphase $JOBCTLO.
4. Besides the above 10 SVA-resident command-processors there are 6 other SVA-resident service routines related to job control.

   $IJBASGN This phase is called in the expansion of the ASSIGN macro. Refer to the *System Macros Reference* manual for a description of the ASSIGN macro.

   $IJBCJC This phase is called in the expansion of the CONDJC macro. Refer to the *IPL and Job Control Diagnosis Reference* manual for a description of the CONDJC macro.

$IJBCJC stores and retrieves conditional job control information (such as ON conditions, last and maximal return code of a job step).

$IJBJCCN This phase is executed during EXEC PGM processing, after $JOBCTLE has completed and before control is being passed to the user program. $IJBJCCN invalidates the partition, loads the user program into the partition (SVC 4) and passes control to the user program (SVC 133). When the user program returns (with BR R14 or EOJ macro), $IJBJCCN is invoked again to handle the return code passed by the user program (via R15 or via the RC operand of the EOJ macro).

$IJBPROC This phase is called in the expansion of the GETSYMB, PARMMAC and PROCMAC macro. Refer to the *System Macros Reference* manual for a description of the GETSYMB macro. Refer to the *IPL and Job Control Diagnosis Reference* manual for a description of the PARMMAC macro.
GETSYMB and PARMMAC provide services to store or retrieve symbolic parameters and their values (for example during SETPARM, PROC, EXEC PROC or EXEC REXX processing). PROCMAC invokes LIBR services to retrieve records from a cataloged procedure (for example during EXEC PROC or EXEC REXX processing).

$IJBSLA This phase is called in the expansion of the LABEL macro. Refer to the *IPL and Job Control Diagnosis Reference* manual for a description of the LABEL macro.
$IJBSLA stores and retrieves label information records. It writes to or reads from the system's label information area.

$IJBSTRT This phase is called in the expansion of the STARTP macro. Refer to the *Supervisor Diagnosis Reference* manual for a description of the STARTP macro.

For these SVA-resident service routines the entry point is identical to the load point, which can be retrieved by means of the LIBR LISTDIR SDL command. Control is passed via BALR/BASSM 14,15. Upon entry, Reg1 points to a parameter list, R14 points back to the caller and Reg0 may contain a function code.
Example: When entering $IJBSLA, Reg1 contains the address of the LPL (label parameter list generated by the LPL macro). Reg0 contains the function code, such as 1 for GETLBL, 4 for ADDLBL or 6 for CLRGRPL etc.

Finally Figure 57 illustrates, which subphases are loaded during execution of a simple job.

```
// JOB EXAMPLE                  (load $JOBCTLG)[1], pass control to $JOBCTLG
// OPTION PARSTD=ADD                              pass control to $JOBCTLG
// DLBL EXAMPLE,'EXAMPLE.FILE'  load $JOBCTLK, pass control to $JOBCTLK
// EXTENT ,VSE200,,,1500,30                       pass control to $JOBCTLK
// OPTION USRLABEL              load $JOBCTLG, pass control to $JOBCTLG
// ASSGN SYSLST,00E             load $JOBCTLD, pass control to $JOBCTLD
// EXEC LSERV,PARM='PARSTD=BG'  load $JOBCTLE, pass control to $JOBCTLE
/&                              load $JOBCTLG, pass control to $JOBCTLG
```

Figure 57. Sample Job to Demonstrate Subphase Loading

Job Control Dumps: Where to Look First?

The term “job control dump” denotes a partition dump taken at a time, when job control (and not a user program) was active in the partition. The most important storage area to look at is the save area located at the very start of the partition.

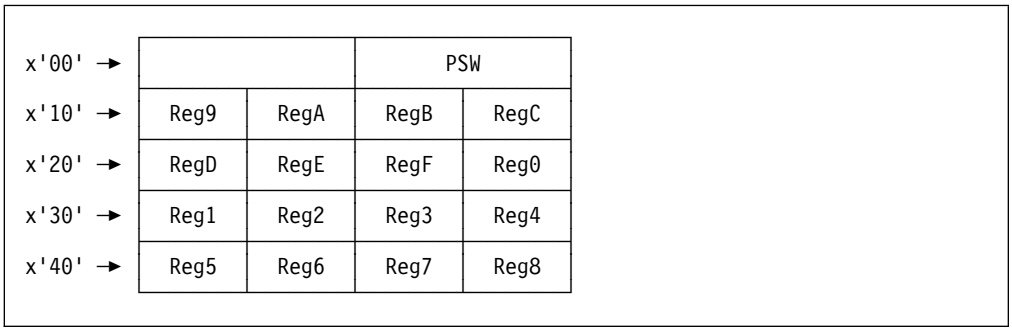


Figure 58. Partition Save Area Layout

For the examples discussed below we assume that the partition starts at X'600000'. When the root phase or one of the subphases is executing, the PSW contains instruction addresses in the range from X'600000' to X'613FFF'. In this case, the following register conventions are in effect:

RegA
Points to the partition's COMREG.

Reg7 Points to a branch vector table located at X'6000E0' in $JOBCTLA. This branch vector table allows the job control subphases to exploit common service routines provided by $JOBCTLA, such as operand scanners, message writers etc. Reg7 is also the first base register of root phase $JOBCTLA.

Reg8 Start of the overlay area (X'609000'), where subphases are loaded. Reg8 is also the first base register of each subphase $JOBCTLx.

Reg7, Reg6, Reg9, RegB
Base registers of root phase $JOBCTLA.

Reg8, Reg9 (, Reg6 (, RegC))
Base registers of subphase $JOBCTLx. It depends on the subphase, if 2, 3 or 4 base registers are needed.

If however one of the SVA-resident command-processors or service routines is executing, the PSW contains instruction addresses from the SVA-24 or SVA-31 area. In this case there are no general register conventions: it is up to the PLX compiler to choose registers.

The next storage area to look at is the eyecatcher at X'78' relative to the partition start address.

[1] At this point a load of subphase $JOBCTLG is not required, if the preceding command (that is the last command in the preceding job) was a /& and hence $JOBCTLG is still in storage.

```
V00600070 40404040 40404040 5BD1D6C2 C3E3D3C1 96 *        $JOBCTLA*
V00600080 5CF5F2C3 5CC4D9C9 E540C7C1 5CF1F161 96 **52C*DRIV GA*11/*
V00600090 F2F761F1 F35CF5F6 F8F660C3 C6F9404D 96 *27/13*5686-CF9 (*
V006000A0 C35D40C3 D6D7E8D9 C9C7C8E3 40C9C2D4 96 *C) COPYRIGHT IBM*
V006000B0 40C3D6D9 D74B40F1 F9F7F76B 40F2F0F1 96 * CORP. 1977, 201*
V006000C0 F1000000 00000000 00000000 00000000 96 *1..............*
```

Figure 59. Eyecatcher of Job Control Root Phase

This eyecatcher contains useful information. Separated by an asterisk we find:

$JOBCTLA In this case this is (per chance) the name of the root phase itself. If a subphase (for example $JOBCTLE) is loaded, the corresponding suffix (in this example E) will overlay the A, thus resulting in $JOBCTLE and reflecting, which subphase is currently active (see Figure 55 on page 109). However the rest of the eyecatcher remains unchanged, that is it applies to $JOBCTLA.

52C This is the release code of z/VSE Version 5, Release 2.
The release code of z/VSE Version 5, Release 1, is 51C.
The release code of z/VSE Version 4, Release 3, is 02C.
The release code of z/VSE Version 4, Release 2, is 01C.
The release code of z/VSE Version 4, Release 1, is 91C.
The release code of z/VSE Version 3, Release 1, is 81C.

DRIV GA This is the service level of the root phase $JOBCTLA and therefore of particular interest for z/VSE Level 2.

11/27/13 This is the date of the last development activity for $JOBCTLA, the date format used is mm/dd/yy.

If job control is in command processing mode (not in initialization mode) the next storage area to look at is the eyecatcher near X'9000' relative to the partition start address. This eyecatcher is preceded by a branch table. Each branch table entry relates to a job control command or to an internal function provided by the subphase. Figure 60 shows 6 branch table entries, the first 5 of which are corresponding to the RSTRT, DLBL, EXTNT, TLBL, and SETPRT commands.

```
V00609000 47F09920 47F091C6 47F08E40 47F08E36 96 *.0r..0jF.0. .0..*
V00609010 47F08EC4 47F062C8 47F0630E 47F09190 96 *.0.D.0.H.0...0j.*
V00609020 47F069B2 47F06AAC 47F07008 47F07008 96 *.0...0...0...0..*
V00609030 5BD1D6C2 C3E3D3C7 5CF5F2C3 5CC4E8F4 96 *$JOBCTLG*52C*DY4*
V00609040 F6F6F4F2 5CF1F161 F0F461F1 F15CF5F6 96 *6642*11/04/11*56*
V00609050 F8F660C3 C6F9404D C35D40C3 D6D7E8D9 96 *86-CF9 (C) COPYR*
V00609060 C9C7C8E3 40C9C2D4 40C3D6D9 D74B40F1 96 *IGHT IBM CORP. 1*
V00609070 F9F7F76B 40F2F0F1 F1D7C1E3 C3C84000 96 *977, 2011PATCH .*
```

Figure 60. Eyecatcher of Job Control Subphase

This eyecatcher contains useful information. Separated by an asterisk we find:

$JOBCTLG
This is the name of the subphase currently active (same as on location X'600078').

52C This is the release code of z/VSE Version 5, Release 2.

DY46642 This is the service level of the subphase $JOBCTLG and therefore of particular interest for VSE Level 2.

11/04/11 This is the date of the last development activity for $JOBCTLK, the date format used is mm/dd/yy.

Another storage location to look at is X'1BC' relative to the partition start address. There we find the command input buffer, showing, which command is currently being executed.

```
V006001B0  006051E4 40404040 40404040 616140C4 16 *.-.U        // D*
V006001C0  D3C2D340 C1C2C36B 7DC1C2C3 4BE3C5E2    *LBL ABC,'ABC.TES*
V006001D0  E34BC6C9 D3C57D6B F2F0F1F4 61F3F6F5    *T.FILE',2014/365*
V006001E0  40404040 40404040 40404040 40404040    *                *
```

Figure 61. Job Control Command Input Buffer

How Job Control Handles Program Return Codes, ABEND and CANCEL Conditions

This chapter discusses the different ways an application program can return to the operating system. Job control can react on these different ways by conditions specified in ON statements (such as $CANCEL, $ABEND, and $RC).

Job control maintains private variables for $RC (the return code of the last job step) and $MRC (the maximal return code of all job steps executed so far). These variables are implemented as 4-byte fields containing either X'40404040' (initial value) or a number (the return code) in character representation. The following job control statements or commands cause an update of these variables:

1. Both JOB and /& statement initialize both variables ($RC and $MRC) to X'40404040'.
2. The SET RCZERO command re-initializes $RC to X'40404040'.
3. The SET RC and SET MRC commands allow to set the (last) return code and the maximum return code.
4. OPTION and STDOPT options ABENDRC / CANCELRC / JCANCLRC may be used to have the system set predefined (last) return codes in cancel situations.
5. The SET MRCZERO command re-initializes $MRC to X'40404040'.
6. The EXEC PGM=pgmname statement re-initializes $RC to X'40404040'. Then it is *up to the application program* to cause an update of $RC and $MRC or not. The return code variables are updated if the application program
   a. returns control to the system by an explicit branch to the return address initially provided in register 14. Then the contents of register 15 is considered as a return code (see *Guide to System Functions*, chapter *Using Conditional Job Control*).
   b. specifies an explicit return code in the RC operand of an EOJ or DUMP macro.

   This behavior is illustrated by the control flow in Figure 62 on page 116.

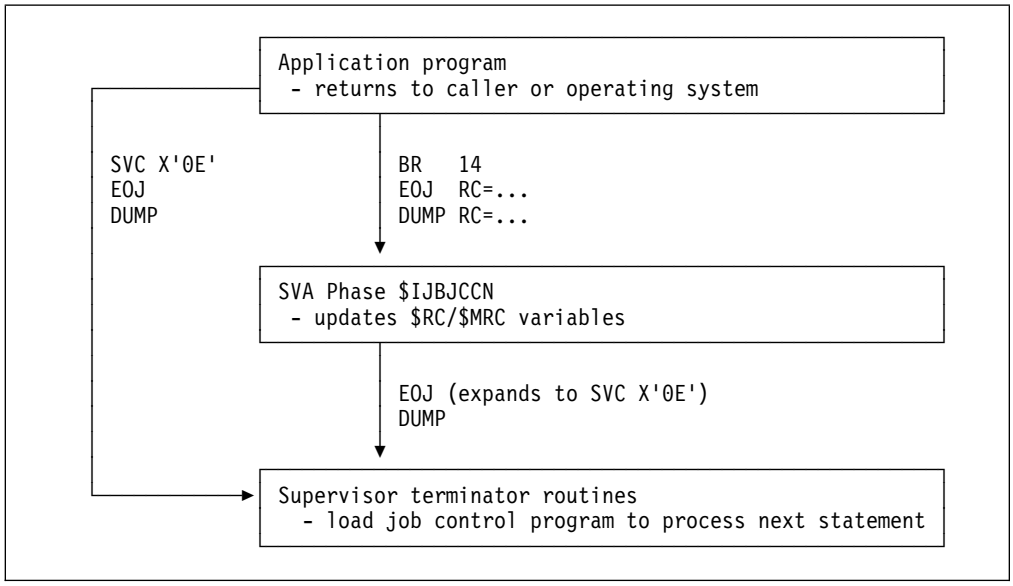


Figure 62. Normal Termination of an Application Program

If the RC operand is omitted in the EOJ or DUMP macro, or if the program returns via SVC X'0E', then there is *no* return code passed to job control, that is $RC retains its initial value X'40404040' and $MRC is not updated at all. For example system utility programs (such as LSERV or LVTOC) return via SVC X'0E' (that is the expansion of the EOJ macro without RC operand) and hence do *not* pass a return code.

At the end of each *job step*, job control checks whether the $RC variable is different from its initial value. If so, the following message is written to SYSLST (unless OPTION NOLOG is in effect):

```
1S55I  LAST RETURN CODE WAS nnnn
```

and nnnn is the contents of the $RC variable.
At the end of each *job*, job control checks whether the $MRC variable is different from its initial value. If so, the following information is part of the end-of-job message:

```
MAX.RETURN CODE=nnnn
```

and nnnn is the contents of the $MRC variable.

Note: Only application programs can cause an update of the $RC/$MRC variable with a *return code*, that is with a value different from the initial value X'40404040'. The job control program itself only initializes the $RC/$MRC variables, any value different from the initial value is stored exclusively on request of an application program.

If $RC and $MRC have their initial values, then the comparisons in

```
IF $RC=0 THEN
IF $MRC=0 THEN
```

are evaluated as true. Also the symbolic parameters RC1 and MRC1 defined by

```
SETPARM RC1=$RC,MRC1=$MRC
```

are assigned the value 0000.

Only programs which terminate normally can cause an update of the $RC/$MRC variables (see Figure 62). Abending programs cannot pass a return code: there is no update of the $RC/$MRC variables and there is no message 1S55I. An application program raises a $ABEND condition, that is terminates *abnormally* for various reasons, the most popular of which are listed below:

1. It produces a program check.
2. It issues a LOAD macro for a phase that could not be found in the LIBDEF phase search chain.
3. It executes an invalid SVC.
4. It executes an SVC 6 (expansion of a CANCEL macro).
5. It produces a LIOCS error (for example EXEC LVTOC, when SYS004 or SYS005 are unassigned or assigned incorrectly).

If a job contains one single job step and this job step terminates abnormally, then the $RC/$MRC variables still have their initial values. Hence there is no message 1S55I on SYSLST and the EOJ message does not contain the MAX. RETURN CODE=nnnn information.

A little confusion is sometimes caused by the following scenario: A job contains two (or more) job steps, the first of which terminates with return code 0 (for example via EOJ RC=0), whereas the second job step terminates abnormally. For the sake of clearness let's consider the following three simple assembler programs:

```
PUNCH ' PHASE SETRC0,S'
SR    15,15       SET RETURN CODE 0
BR    14          BACK TO CALLING PROGRAM $IJBJCCN
END

PUNCH ' PHASE PGMCHECK,S'
DC    XL2'0000'   CAUSE PROGRAM CHECK
END

PUNCH ' PHASE SETRC333,S'
LA    15,333      SET RETURN CODE 333
BR    14          BACK TO CALLING PROGRAM $IJBJCCN
END
```

The sample job

```
// JOB SAMPLE1
// EXEC SETRC0    CAUSE UPDATE $RC=0000 and $MRC=0000
// EXEC PGMCHECK  CAUSE PROGRAM CHECK
/&
```

will display the following, at first glance contradictory messages:

```
1S78I  JOB TERMINATED DUE TO  PROGRAM ABEND
EOJ SAMPLE1   MAX.RETURN CODE=0000
```

Program PGMCHECK did not terminate normally, and hence did not cause an update of $RC/$MRC. At end-of-job time $RC contains its initial value (from EXEC processing) and $MRC contains 0000 (from EXEC SETRC0), which is displayed in the end-of-job message. If you want your end-of-job message to reflect program abends by a special return code (for example 0333), you can modify job SAMPLE1 as follows.

```
// JOB SAMPLE2
// ON $ABEND GOTO ABNORMAL
// ON $RC=333 CONTINUE
// EXEC SETRC0    CAUSE UPDATE $RC=0000 and $MRC=0000
// EXEC PGMCHECK  CAUSE PROGRAM CHECK
// GOTO ENDOFJOB
/. ABNORMAL
SET MRCZERO       RE-INITIALIZE $MRC
// EXEC SETRC333  CAUSE UPDATE $RC=0333 and $MRC=0333
/. ENDOFJOB
/&
```

In case of erroneous job control, but also in case of program malfunction (for example a loop) the operator may want to cancel a job. This can be achieved by the following commands:

1. Job control `CANCEL` command: only applicable if the partition is waiting for an operator response.
2. Attention routine `CANCEL partition` command.
3. POWER `PFLUSH partition` command for POWER-controlled partitions.

If a job step is canceled via AR CANCEL or POWER PFLUSH, the job step is not able to cause an update of the $RC/$MRC variables. The maximum return code eventually displayed as part of the end-of-job message applies to the return codes set by preceding job steps (if any), not to the job step being canceled. So the messages

```
1S78I  JOB TERMINATED DUE TO  CANCEL COMMAND
EOJ LOOP      MAX.RETURN CODE=0000
```

indicate, that before the CANCEL or PFLUSH command was executed there was at least one job step which completed successfully and had $RC/$MRC set to 0000.

Similar to job SAMPLE2 you can specify an `ON $CANCEL GOTO label` to handle an operator's CANCEL or PFLUSH.

Notes:

1. No matter, whether a job step terminated normally or abnormally or whether it was canceled by the operator: the job accounting routine $JOBACCT (see *Guide to System Functions* manual) is executed in any case. The cancel code (part of the job accounting table) indicates, whether the job step terminated normally or not. For cancel codes see chapter *VSE/Advanced Functions Cancel Codes* in the *Messages and Codes* manual.
2. Conditions which cause (normal or abnormal) termination of an application program can be handled by exit routines established by STXIT macros (see *System Macros Reference* manual for details). For example, program PGMCHECK from above can be modified to simply ignore the program check (operation exception):

```
         PUNCH ' PHASE PGMCHECK,S'
         BASR  4,0                 ESTABLISH ADDRESSABILITY
         USING *,4
         STXIT PC,PCRTN,PCSAV      SET UP PROGRAM CHECK LINK
         DC    XL2'0000'           CAUSE PROGRAM CHECK
         EOJ                       NORMAL TERMINATION
PCRTN    BASR  4,0                 ESTABLISH ADDRESSABILITY
         USING *,4
         WTO   '*** PC EXIT: PROGRAM CHECK IS IGNORED.'
         EXIT  PC                  RETURN FROM EXIT ROUTINE
PCSAV    DS    (SVULNGTH)X'00'     SAVE AREA
         MAPSAVAR                  MAP SAVE AREA
         END
```

In this case, the jobs SAMPLE1 or SAMPLE2 terminate normally and the action specified in an ON $ABEND statement is not carried out.

Many IBM and vendor programs use STXIT to establish AB or PC exits. For example the Language Environment (LE/VSE) handles conditions raised by program interrupts or abends, provided the runtime option TRAP(ON) is in effect. If a program causes a program check in the LE runtime-environment, message CEE3321C is displayed rather than message 0S03I. For details see chapter *Understanding Abend Codes and VSE Cancel Codes* in the *LE/VSE Debugging Guide and Run-Time Messages* manual.

Default ON Conditions in Conditional Job Control

A common misunderstanding occurs when checking $RC with the IF statement.

Program SETRC333 (see page 117) sets a return code of 333. Consider the following job:

```
// JOB IFSAMPLE
// EXEC SETRC333  CAUSE UPDATE $RC=0333 and $MRC=0333
// IF $RC = 333 THEN
// PAUSE We got return code 333.
/&
```

Users not familiar with the default ON conditions might expect that the PAUSE statement is executed, because the condition in the preceding IF statement is evaluated as true. However not only the PAUSE statement but also the IF statement itself are ignored, because the default ON condition `ON $RC >= 16 GOTO $EOJ` is in effect.

When SETRC333 terminates, the terminator routine $IJBSEOT will load the job control root phase $JOBCTLA into the partition. $JOBCTLA will check existing ON conditions. Because the condition `$RC >= 16` is obviously met, the corresponding action `GOTO $EOJ` is taken. The following IF and PAUSE statements are skipped.

Job Control Error Handling

This chapter discusses job control errors and related messages, how they are affected by certain options (such as ACANCEL, JCANCEL, and SCANCEL) and how they can be trapped with ON $CANCEL.

Job control error messages have prefix 1 followed by a letter other than H, I, P, Q, R, and V. They may have different action indicators, such as I (Information), D

(Decision) or A (Action). One and the same message can appear with different action indicators. The action indicator depends on

1. the way the message is internally defined.
2. certain options (such as ACANCEL, JCANCEL, and SCANCEL). Unless otherwise stated we assume that the default options (NOACANCEL, NOJCANCEL and NOSCANCEL) are in effect.
3. whether the erroneous statement was read from SYSRDR or from SYSLOG.

There is a minority of job control messages that appear as I-type only: for example message 1S55I on SYSLST (provided OPTION LOG is in effect) and on SYSLOG, if the corresponding EXEC statement was issued from SYSLOG. Another example is message 1U76I in the output of the JCLEXIT command, which is written to SYSLOG, not to SYSLST. These pure I-type messages appear as responses to *valid* commands (such as EXEC or JCLEXIT in the examples above). If OPTION NOLOG is in effect, neither the valid command nor the subsequent I-type message is written to SYSLST.

There is another minority of job control messages that appear as D-type only, for example message 1I00D (READY FOR COMMUNICATIONS).

The majority of job control messages appear as both I-type and D-type (or A-type) message. These kind of messages appear as responses to *invalid* commands, therefore they are *error messages*. Both the invalid command and the subsequent error message are written to SYSLOG and SYSLST, regardless whether LOG and OPTION LOG are in effect or not. If an error message appears as I-type message, the job is cancelled, if it appears as D-type (or A-type) message, the system waits for an operator response. Let's consider two examples:

```
// JOB EXAMPLE1                                          // JOB EXAMPLE2
// EXEC LSERV,SIZE=64K,PARM='PARSTD=BG',GRACE           // EXEC NOTTHERE
/&                                                       /&
```

Job EXAMPLE1 will issue a D-type message

```
1S08D  INVALID STATEMENT.
```

and the fourth digit in the message identifier is pointing to the operand in error. In this case it is the 8th operand, that is the GRACE operand which should have been spelled as TRACE. When using this field count, you must start counting with 1 (for the // at the very left side of the statement) and move to the right by incrementing 1 when one of the following field separators (so-called scan stop characters) appears: blank, comma or equal sign. Items enclosed within apostrophes (for example 'PARSTD=BG') are counted as *one* field, even if they contain field separators, such as the equal sign in 'PARSTD=BG'. For more details see the beginning of chapter *1-Prefix z/VSE Messages* in the *Messages and Codes* manual.
1S0nt is one of the messages which are internally defined as D-type messages. This is because they relate to errors which are supposed to be corrected by the operator easily and immediately.

We assume that phase NOTTHERE is not in the LIBDEF phase search chain. Then job EXAMPLE2 will issue an I-type message

```
1U53I  PROGRAM NOT FOUND.
```

and job EXAMPLE2 is cancelled, that is the rest of the job's statements are flushed.

1U5nt is one of the messages which are internally defined as I-type messages and raise an internal cancel condition. This is because they relate to severe errors which are not only caused by the one preceding statement. In job EXAMPLE2 message 1U5nt can be caused by a typo, by a missing LIBDEF statement or by a programmer who forgot to catalog NOTTHERE. It is unlikely that an operator can determine the reason of the error message and correct the problem.

In case of multiple job steps, message 1U5nt and the subsequent messages can cause some confusion. If the first job steps completed successfully and passed return codes of 0 (see “How Job Control Handles Program Return Codes, ABEND and CANCEL Conditions” on page 115), then the failing EXEC NOTTHERE job step will produce the following output in job EXAMPLE3 (for program SETRC0 see page 117):

```
// JOB EXAMPLE3
// EXEC SETRC0    CAUSE UPDATE $RC=0000 and $MRC=0000
// EXEC NOTTHERE
/&

1U53I  PROGRAM NOT FOUND.
1I70I  JOB EXAMPLE3 CANCELLED DUE TO CONTROL STATEMENT ERROR
1S78I  JOB TERMINATED DUE TO  PROGRAM ABEND
EOJ EXAMPLE3  MAX.RETURN CODE=0000
```

The second job step in EXAMPLE3 caused an I-type job control error message. Program NOTTHERE could not be executed, and hence could not cause an update of $RC/$MRC. At end-of-job time $RC contains its initial value (from EXEC processing) and $MRC contains 0000 (from EXEC SETRC0), which is displayed in the end-of-job message. If you want your end-of-job message to reflect job control problems by a special return code (for example 0333, compare page 117) you can modify job SAMPLE3 as follows. Note that you need to specify the SCANCEL option to have the internal cancel condition trapped by ON $CANCEL.

```
// JOB EXAMPLE3
// ON $CANCEL GOTO JCLERROR
// ON $RC=333 CONTINUE
// OPTION SCANCEL TRAP JCL INTERNAL CANCEL CONDITIONS
// EXEC SETRC0    CAUSE UPDATE $RC=0000 and $MRC=0000
// EXEC NOTTHERE
// GOTO ENDOFJOB
/. JCLERROR
SET MRCZERO       RE-INITIALIZE $MRC
// EXEC SETRC333  CAUSE UPDATE $RC=0333 and $MRC=0333
/. ENDOFJOB
/&

1U53I  PROGRAM NOT FOUND.
1I70I  JOB EXAMPLE3 CANCELLED DUE TO CONTROL STATEMENT ERROR
EOJ EXAMPLE3  MAX.RETURN CODE=0333
```

So job control error messages may appear as I-type messages (serious error causing an internal cancel condition) or as D-type messages (easy to correct error waiting for operator response).

If you run in an unattended environment, you may want all job control error messages to appear as I-type messages. This can be achieved by specifying the JCANCEL option. If option JCANCEL were in effect for job EXAMPLE1, then message 1S0nt would appear as I-type and the job would be cancelled. If both

JCANCEL and SCANCEL are in effect, then you can trap any job control error with an ON $CANCEL condition. While option JCANCEL converts *all* D-type error messages into I-type, option ACANCEL converts only those D-type messages, that are caused by incorrect ASSGN or LIBDEF statements.

On the other hand, you may want all job control error messages to appear as D-type (or A-type) messages, thus waiting for operator response. To achieve this, OPTION NOJCANCEL (the default) is a necessary condition, but not sufficient. You need to specify both NOJCANCEL and the LOG command. If you want to determine the origin of a JCL error message, which would have been displayed as I-type and would have caused an internal cancel condition, the preceding statement is not enough: you need all the job's statements to find out what went wrong or what is missing. That's why the LOG command is a reasonable prerequisite to convert I-type messages to D-type (or A-type). Let's consider the following console output:

```
// JOB EXAMPLE4
DATE 03/10/2000, CLOCK 09/08/26
// EXTENT SYS005,VOL123,4,0,150,15
1L05I  INVALID LABEL SYNTAX.
1I70I  JOB EXAMPLE4 CANCELLED DUE TO CONTROL STATEMENT ERROR
1S78I  JOB TERMINATED DUE TO  PROGRAM ABEND
EOJ EXAMPLE4
```

The EXTENT statement is syntactically correct, but nevertheless it complains about the 5th field (extent type 4). The origin of the problem (and the error message) is that the DLBL and the EXTENT statement do not match: DLBL omitted the label type, so the default SD was assumed. EXTENT however tried to specify information for an ISC or ISE label type. This example shows that you may need more than the last statement to find out the source of the problem. So option NOJCANCEL together with the LOG command allows to debug job control statements by displaying the whole sequence of statements on SYSLOG:

```
// JOB EXAMPLE4
DATE 03/10/2000, CLOCK 09/49/11
LOG
// DLBL SAMPLE,'SAMPLE.FILE'
// EXTENT SYS005,VOL123,4,0,150,15
1L05D  INVALID LABEL SYNTAX.
```

Job Control Statement Logging

This chapters describes job control statement logging on a plain-vanilla VSE system. OEM software may enable JCL exit routines as well as vendor exit routines which change the default way of logging statements on SYSLST or SYSLOG.

In the text which follows it is assumed that SYSLST is assigned.

Job Control Statement Logging on SYSLST

Statement logging on SYSLST is controlled by the following job control statements:

- // STDOPT LOG=YES|NO for the overall system - permanent option
- // OPTION LOG|NOLOG for each individual job - temporary option
- // LOG or // NOLOG for each individual job, which is equivalent to // OPTION LOG or // OPTION NOLOG respectively.

Per default the permanent LOG option is set to YES, that is *all* job control statements are logged on SYSLST. If the permanent LOG option is set to NO, only *invalid* job control statements are logged on SYSLST.

The LOG option provided in an OPTION statement overrides the permanent STDOPT value temporarily. It is reset to the permanent STDOPT value during EOJ processing.

Note: The LOG option only controls, whether job control statements are written to SYSLST. It does not affect program output to SYSLST. For example, if your job stream contains // EXEC LSERV, then the LSERV utility output is *always* written to SYSLST, whether the LOG option is in effect or not. The LOG option however controls whether the // EXEC LSERV statement itself is written to SYSLST or not.

Job Control Statement Logging on SYSLOG

Statement logging on SYSLOG is controlled by the job control or attention routine LOG command.

Note: Do not mistake the LOG command for the // LOG statement. The LOG command controls logging on SYSLOG while the // LOG statement controls logging on SYSLST.

After the LOG command has been specified, *all* job control statements are written to SYSLOG. The LOG command remains in effect, until a subsequent NOLOG command is given. The NOLOG command does not prevent *all* job control statements from being written to SYSLOG. Some statements, which are considered to be important for the operator, are logged unconditionally on the console, such as

- // JOB statement or EOJ message caused by /&-processing.
- The * (comment) statement.
- Commands affecting the partition layout such as ALLOC or SIZE.
- Commands affecting the I/O system such as DVCUP.
- Invalid commands.

Note: There are several job control commands displaying various system information, such as MAP, QUERY, LISTIO, JCLEXIT, or DVCDN. The output of these commands (not necessarily the command itself) is always displayed on SYSLOG, even if the NOLOG command is in effect.

Job Control Comment Statements

The job control comment statement begins with the symbol '*'.

```
* This comment is always displayed on SYSLOG.
```

It is *always* written to SYSLOG, whether LOG was specified or not. The comment statement is appropriate to inform the operator about a job's purpose. If followed by a PAUSE statement, the * statement can be used to request operator action.

Programmers who write job control may want to document their JCL by comments. Such internal comments should never be displayed on SYSLOG or SYSLST, so the comment statement is not suitable. The /* statement (end-of-data indicator for programs reading from SYSRDR or SYSIPT) however can be used to insert internal comments into jobs.

```
/* This comment is never displayed, neither on SYSLOG nor on SYSLST.
```

Such kind of comments must not be inserted into input data read by programs.

If you want comments to be displayed dependent on the LOG command (SYSLOG) or the LOG option (SYSLST), then the /. label statement can be a good choice. Provided that CMT is a label never referenced by any GOTO, then you can insert comments as follows:

```
/. CMT This comment is displayed dependent on LOG command/option.
```

Figure 63 summarizes the different comments and shows whether they are displayed on SYSLOG or SYSLST.

Figure 63. Logging of Comment Statements

	Logged on SYSLOG?	**Logged on SYSLST?**
* Comment	always	dependent on LOG option
/* Comment	never	never
/. CMT Comment	dependent on LOG command	dependent on LOG option

Note: If job control is in skip mode, that is a GOTO statement is being executed, then all of the above mentioned comments are skipped.

VSE Label Area – Layout and Capacity Considerations

This section presents answers to frequently asked questions, such as:

- How many labels will fit in my label area?
- How can I tell how much space is left in my label area?

Before providing answers to the questions listed above, I need to explain some basic design concepts and introduce three important terms:

1. Label Information Record (abbreviated as LIR),
2. Label Area Segment (abbreviated as LAS),
3. Label Group (abbreviated as LG), which is sometimes called label subarea.

If you are not interested in technical background information, the following three sections on LIRs, LASs and LGs may be boring rather than fun. You can skip them and go to “Frequently Asked Questions” on page 130 right away. If, however, you'd like to get an idea on what's inside label processing, I encourage you to continue reading.

Label Information Record (LIR)

An LIR is the smallest unit of information contained in the label area. Its length varies from a minimum of 32 bytes to a maximum of 5204 bytes.

While a TLBL statement *always* produces one single 80-byte LIR, things are more complicated with the DLBL statement. A DLBL-EXTENT statement sequence can produce one or more LIR(s) with different lengths, depending on the:

- File type code (SD, DA, ISC, ISE, VSAM) in the DLBL and
- Number of EXTENT statements following the DLBL statement.

Consider the following sample DLBL-EXTENT statement sequences:

```
*  SAMPLE1                              *  SAMPLE2
// DLBL FILNAME,'FILE ID',2002/365,SD  // DLBL IJDFILE,'POWER.DATA.FILE',99/365,DA
// EXTENT SYS005,000020,1,0,10,2010    // EXTENT SYS002,,1,0,15,90
// EXTENT SYS005,000020,1,1,4000,1510  // EXTENT SYS002,,1,1,105,60
// EXTENT SYS006,000100,1,2,64,1300    // EXTENT SYS003,,1,2,165,90
*  No more EXTENT statements follow    *  No more EXTENT statements follow
```

SAMPLE1 produces 3 LIRs, each 104 bytes long; while SAMPLE2 produces 1 LIR, 144 bytes long. Why?

Basically it takes 84 bytes to store the information contained in one DLBL card and 20 bytes to store the information in one EXTENT card. How these 84-byte and 20-byte chunks are composed to make up an LIR depends on the file type code (and its access method). For non-sequential files (file type codes other than SD), a DLBL-EXTENT statement sequence produces one single LIR with a length of 84+(**n***20) bytes, where **n** denotes the number of EXTENT statements.

For sequential files (file type code SD), a DLBL-EXTENT statement sequence produces **n** LIRs with a length of 84+20=104 bytes each, where again **n** denotes the number of EXTENT statements.

Figure 64 shows the different kinds of LIR-producing job control statements and the number and length of LIRs being produced.

Figure 64. Different Lengths of Label Information Records

TLBL/DLBL-EXTENT Statement	**Corresponding LIR(s)**
TLBL	80-byte LIR
DLBL...SD – no EXTENT statement following	84-byte LIR
DLBL...DA|ISC|ISE – no EXTENT statement following	invalid
DLBL...VSAM – no EXTENT statement following	84-byte LIR plus one additional 32-byte LIR[2]
DLBL...SD – n EXTENT statements following	n LIRs, 104-byte each
DLBL...DA|ISC|ISE – n EXTENT statements following	1 LIR, 84+n*20 bytes long
DLBL...VSAM – n EXTENT statements following	(84+n*20)-byte LIR plus one additional 32-byte LIR[2]

If you want to know how many LIRs are currently stored in your label area, you can write a program invoking appropriate LABEL macro requests. Or you simply have a REXX procedure inspect the output of the LSERV utility:

[2] only if VSAM-specific operands such as DISP, RECORDS, RECSIZE, BUFND, or BUFNI are specified

```
/* REXX                                                              */
i=0                                     /* Provide job input stem    */
i=I+1; cd.i = '* $$ JOB JNM=LSERV,CLASS=C,LOG=NO'
i=I+1; cd.i = '// JOB LSERV'
i=I+1; cd.i = '// EXEC LSERV'
i=I+1; cd.i = '/&'
i=I+1; cd.i = '* $$ EOJ'
cd.0 = i
ADDRESS POWER                           /* Have POWER execute job stem  */
"PUTQE RDR JOBNAME LSERV JOBNUM JOBNR WAIT 500 STEM cd."
"GETQE LST JOBNAME LSERV JOBNUM" JOBNR "STEM lserv."
tlblind  = 'GENERATION NUMBER'          /* TLBL card identifier      */
dlblind  = 'FILE TYPE'                  /* DLBL card identifier      */
extntind = 'EXTENT SEQUENCE NUMBER'     /* EXTENT card identifier    */
extnthdr = 'EXTENT INFORMATION                          OMITTED'
vsamind  = 'ADDITIONAL INFORMATION'     /* 2nd LIR for type code VSAM   */
sequind  = 'SEQUENTIAL'                 /* file type code SD         */
bytes = 0; lirs = 0; dlbls = 0; tlbls = 0    /* initialize counts    */
do i=1 to lserv.0                       /* examine LSERV output lines   */
  select                                /*                           */
    when pos(tlblind,lserv.i) = 11 then do   /* TLBL identifier in column 11 */
      bytes = bytes+80+2                /* TLBL LIR is 80 + length info */
      tlbls = tlbls+1; lirs = lirs+1; end    /* Increment TLBL and LIR count */
    when pos(dlblind,lserv.i) = 11 then do   /* DLBL identifier in column 11 */
      dlbls = dlbls+1                   /* Increment DLBL count      */
      if pos(sequind,lserv.i) = 56 then      /* file type code SD         */
        flag = 0                        /* set switch for EXTENT info   */
      else do                           /*                           */
        flag = 1                        /* set switch for EXTENT info   */
        bytes=bytes+84+2; lirs = lirs+1; end /* DLBL card info is 84...   */
    end                                 /* ...plus 2 byte length info   */
    when (pos(extnthdr,lserv.i) = 6),       /* EXTENT information omitted...*/
       & (flag = 0) then do                 /* ...for SD file            */
      bytes = bytes+84+2; lirs = lirs+1; end /* SD file without EXTENTs   */
    when pos(vsamind,lserv.i) = 6 then do    /* 2nd LIR for file type code...*/
      bytes = bytes+32+2; lirs = lirs+1; end /* ...VSAM is 32 bytes long  */
    when pos(extntind,lserv.i) = 11 then do  /* EXTENT identifier column 11  */
      if flag = 1 then                  /* EXTENT info for non-sequentl */
        bytes = bytes + 20              /* add 20 bytes for each EXTENT */
      else do                           /*                           */
      bytes=bytes+104+2; lirs = lirs+1; end  /* SD duplicates DLBL info   */
    end                                 /*                           */
    otherwise nop                       /*                           */
  end                                   /*                           */
end                                     /*                           */
say 'Number of LIRs currently stored in label area:' lirs
say 'Bytes needed to store all LIRs  :' bytes
say 'DLBL statements currently active:' dlbls
say 'TLBL statements currently active:' tlbls
say 'Average number of DLBL/TLBLs accommodated in one LAS:' ,
2048 % ((bytes % (dlbls+tlbls))+1)
```

Figure 65. REXX Procedure for Inspecting LSERV Utility Output

Notes:

1. The number of labels displayed in the DITTO DLA (Display Label Area) function is *not* the number of LIRs. DITTO DLA does not consider multiple LIRs produced by one single DLBL-EXTENT statement sequence. For example, DITTO counts both the label information for FILNAME and IJDFILE as 1. See SAMPLE1 and SAMPLE2 on page 125.

Also, the label number displayed by DITTO DLA is not necessarily the number of DLBL and TLBL statements submitted. If the filenames used in your DLBL/TLBL statements are unique per label group, then both numbers match. If, however, you submit duplicate label information for one and the same filename, such as:

```
// TLBL TAPEFIL,'TAPE.OUTPUT.FILE'       (might be hidden in a PROC)
// TLBL TAPEFIL,'TAPE.OUTPUT.FILE',30
```

then DITTO will count them as one. Note that OPEN processing would pick up the first LIR for TAPEFIL, resulting in a 0-day retention period.

2. Neither LSERV output nor DITTO DLA output contain information on free-usage labels. If you have vendor software installed which exploits free-usage labels, the total amount of LIRs in your label area may be higher than the number displayed by the REXX procedure.

   Starting with z/VSE 3.1 LSERV provides two new parameters, ALL and FREE, to display information on free-usage labels as well. For details see the Diagnosis Tools manual.

3. When an LIR is stored in an LAS, then it is always preceded by an additional two-byte length field. That's why we need to add 2 when incrementing the bytes count in the REXX procedure.

Label Area Segment (LAS)

I assume you use the default VDISK command to define the label area in native data space:

```
VDISK UNIT=FDF,BLKS=2880,VOLID=VDIDLA,USAGE=DLA
```

In this case, the volume layout of VDIDLA is as follows:

Blocks 0000 through 0001: VOL1 label
Blocks 0002 through 2871: File DOS.LABEL.FILE.ON.VIRTUAL.DISK
Blocks 2872 through 2879: VTOC

Only 2870 512-byte blocks are available to store LIRs, because 10 blocks are needed to store VOL1 label and VTOC. These 2870 blocks build a contiguous piece of virtual storage. You can think of it byte-wise, block-wise, or *LAS-wise*.

An LAS is a 2KB chunk of virtual storage starting on a block boundary. It is the physical allocation unit to store LIRs. LAS 0 comprises blocks 2 through 5; LAS 1 comprises blocks 6 to 9; and so forth. The last LAS available is LAS 716, comprising blocks 2866 through 2869. Two blocks – 2870 and 2871 – remain unused. Phase $IJBSLA “reads” and “writes” label data LAS-wise[3].

The SIR command output contains an additional line on LAS consumption. It looks like:

```
LBL.-SEGM.= 00009           HIGH-MARK = 00026    MAX = 00717
```

The first number (9) is the number of LASs currently in use. Note that LAS 0 is always in use by the system. Thus the number of LASs currently in use is always greater than or equal to 1. The second number (26) is the maximum number of

[3] Actually it uses the MVCL instruction to move 2KB-pieces of data from or to the data space.

LASs used since the last IPL. The third number (717) is the number of LASs you made available with the VDISK command discussed above.

From a program, you can retrieve these three values by means of the new CPCTYLBL function of the LABEL macro. For details, see *IPL and Job Control DRM* on the z/VSE Collection DVD (November 2011 or later).

If the second value (HIGH-MARK) is much smaller than the third value (MAX), then you should decrease the BLKS value in the VDISK command instead of wasting virtual storage. If, for example, you find HIGH-MARK to be always less than 400, then a BLKS value of 1920 (corresponding to MAX = 00477) will be sufficient.

On the other hand, if you find HIGH-MARK coming very close to MAX, then you risk that some batch job(s) hang or cancel with message 1L1nD (LABEL AREA EXHAUSTED). It is not possible to increase the size of the label area. The default size (BLKS=2880) is already the maximum size. If you chose a higher BLKS value (for example, BLKS=3840), then the volume layout is as follows:

Blocks 0000 through 0001: VOL1 label
Blocks 0002 through 2880: File DOS.LABEL.FILE.ON.VIRTUAL.DISK
Blocks 2881 through 3831: Free
Blocks 3832 through 3839: VTOC

This is because label processing is currently restricted to deal with a maximum of 720 LASs (corresponding to a bitmap with 720/8 = 90 bytes). This is a compromise between the label area capacity on the one hand and system GETVIS consumption on the other. The more LASs, the longer the bitmaps.

These bitmaps map the physical organization of the label area (LASs) to its logical organization (label groups). Each non-empty label group has a bitmap of its own, reflecting which LASs belong to it.

Label Group (LG)

Most label area display utilities (such as LSERV or DITTO DLA) list labels ordered by LG, such as *class C labels* or *F2 temporary partition labels* or *system labels*. The OPTION statement and its operands (STDLABEL, CLASSTD, PARSTD, USRLABEL) determine into which LG the LIRs created by subsequent DLBL/TLBL statements go.

Think of LGs as a set of empty buckets. There are quite many of them:

- 1 bucket for internal usage (for OPTION...=DELETE processing)
- 1 bucket for system LG
- 23 buckets for class LGs
- Up to 636 buckets for partition LGs. This is because each partition may require 3 LGs (free-usage, temporary, and permanent partition LG), and you can have up to 212 partitions concurrently active, depending on the NPARTS operand of the SYS command in your IPL procedure.

As long as these buckets are empty, they will not use any system resources. When LIRs are to be put into an empty bucket, then

1. One or more LASs are allocated to store the LIRs, and
2. A 97-byte chunk of storage is required to store the bitmap and other control information.

Label processing reserves this storage for the system bucket, the 23 class buckets and the 36 static partitions buckets. It also provides a pool of 27 chunks for dynamic partition buckets.

When more than 27 dynamic partition buckets are active, then the storage requests are satisfied by system GETVIS (112 bytes, LOC=ANY).

Now we have all pieces together. DLBL/TLBL statements create LIRs. LIRs are stored in LASs, which make up the physical label area. LASs are allocated to the LG determined by the OPTION statement (STDLABEL, CLASSTD, PARSTD, or USRLABEL) preceding the DLBL/TLBL. To illustrate, let's examine the very first statements in the BG startup procedure.

When the `VDISK UNIT=FDF,BLKS=2880,VOLID=VDIDLA,USAGE=DLA` command is exectuted during BG startup, all buckets are empty. LAS 0 is reserved for system use, and LAS 1 through 716 are available to store LIRs.

The `EXEC PROC=STDLABEL` statement is processed next, with `// OPTION STDLABEL` at the very top of STDLABEL.PROC. At this point, all following label information will be directed into the system LG, which is still empty. That is, it has no LASs allocated to it.

As soon as the first DLBL-EXTENT statement sequences and TLBL statements are processed, LAS 1 is allocated to store the corresponding LIRs. When LAS 1 is full, LAS 2 is allocated, and so on.

`EXEC PROC=STDLABUP` is executed next. Since OPTION STDLABEL is still in effect, all LIRs are stored in LASs allocated to the system LG.

On the z/VSE 5.2 system as shipped by IBM, it takes 5 LASs to store all system standard labels. Procedure STDLABUS is executed next, which only contains OPTION statements (no DLBL or TLBL follows). Thus the corresponding LG buckets remain empty and have no LASs allocated to them. Finally, the last statement in STDLABEL.PROC is executed, which is a `// OPTION USRLABEL` to have any subsequent LIRs stored in BG's temporary partition LG instead of in the system LG.

When foreground or dynamic partitions get active and they submit DLBL or TLBL statements, then a free LAS will be allocated to the individual partition's LG (temporary or permanent) to store the corresponding LIRs.

Frequently Asked Questions

How many labels will fit into my label area?

It depends on:

- Label area size (BLKS operand in VDISK...USAGE=DLA command).

  The default is 717 LASs or 1434K.

- The kind of labels you deal with – tape labels (TLBL) or disk labels (DLBL).

  For disk labels, the number and length of the LIR(s) depend on the file type code (SD, DA, DU, ISC, ISE, VSAM) and the number of EXTENTs. If multi-extent files are the exception (and not the rule), then assume an average of 128 bytes per DLBL-EXTENT statement sequence or TLBL as a rule of thumb.

  With this assumption, an LAS can hold the LIRs corresponding to 16 DLBL-EXTENT statement sequences or TLBLs. If you had only the system LG and all 717 LASs allocated to it, then the default label area would hold all LIRs corresponding to 717*16 = 11472 DLBL-EXTENT statement sequences or TLBL statements.

  Our assumption that an LAS contains the LIRs corresponding to 16 DLBL-EXTENT statement sequences or TLBL statements must not be true for your site. A more reliable estimate is the number **k**, which is displayed by the the last “say” instruction in the REXX procedure[4]. On the z/VSE 5.2 system as shipped by IBM, this number is 19 rather than 16.

- The number of non-empty LGs, especially those belonging to dynamic partitions.

  Suppose you have class P enabled with maximal 32 dynamic partitions. Further suppose all 32 class P partitions are active and submit one single DLBL/EXTENT sequence for a sequential file. This will allocate 32 LASs (one for each temporary partition LG), each one containing one single LIR.

  If you submitted the 32 DLBL/EXTENT sequences to the class P LG, then 2 LASs would be enough to store 32 LIRs. In both cases, the label area contains the same 32 LIRs. In the first case, they occupy 32 LASs; in the second case, only two!

  The situation can be even worse if you have multiple VSE jobs in one VSE/POWER job, with the need for permanent dynamic partition labels or a vendor product active with the need of free-usage labels. As a rule of thumb, your label area should accommodate at least the LIRs created by:

  ```
  (717 - i*NPARTS - 25) * k
  ```

  DLBL-EXTENT statement sequences or TLBL statements.

  k denotes the average number of DLBL/TLBL statements covered by one LAS, as displayed by the REXX procedure[4].

  i is the average number of non-empty LGs associated with each partition. Set i to 0, when you deal exclusively with system and class labels (no partition labels). Set i to 1 if each partition will submit label information in only 1 LG

[4] See “Label Information Record (LIR)” on page 124.

(either free-usage, temporary, or permanent partition LG) and to 2 or 3 respectively. Your LSERV output can give you a rough guess which value for i is adequate for your site.

In the z/VSE 5.2 system as shipped by IBM, k is 19, i is 1 (because I only deal with temporary partition labels), and NPARTS is 20 (the default). Thus you can accommodate label information created by approximately 12700 DLBL-EXTENT statement sequences or TLBL statements.

How can I tell how much space is left in my label area?

With the additional line in the SIR command's output[5], the answer to this question is trivial. So let's discuss the question *What can I do when there is almost no space left in my label area?* An answer to this can be found by inspecting the LSERV output at peak times.

There are basically two approaches in order to free LASs.

1. If you find not so many labels (well below 10000) contained in many (more than 100) LGs, then the bottleneck might be caused by many (dynamic) partitions submitting label information into probably different partition LGs.

   Try to re-arrange LGs. That is, try to exploit the class LG instead of using permanent and temporary partition LG for each single dynamic partition belonging to this class. Re-arranging LGs is always recommended when you have many non-empty dynamic partition LGs containing only few (5 or less) labels.

   A general recommendation – especially when enabling all 23 classes available in z/VSE 5.2 – is to store label information in the class LG, rather than in each individual partition's permanent or temporary LG.

2. If you find many labels (10000 or more) contained in few (less than 50) LGs, then re-arranging the LGs won't help much. You should inspect label-submitting jobs or PROCs and check whether all this label information – especially for multi-extent sequential files – is required all the time or only for one particular job step.

   In terms of LAS consumption, it is always better to provide temporary label information for each job step, rather than for the whole job. This may reduce the number of LIRs and free LASs, unless you really need to process thousands of files at a time.

   To illustrate, consider a job with 20 job steps, each step dealing with five step-specific disk files, and thus requiring five DLBLs. If your job first submits 100 DLBL statements, followed by 20 EXEC statements, then the temporary partition LG may need seven LASs. If, however, five step-specific DLBL statements precede each EXEC statement, then one single LAS is enough.

   The same is true if you consider a VSE/POWER job containing multiple VSE jobs. In terms of LAS consumption, it is always better to put VSE job-specific label information in each individual job, rather than putting all label information in the first VSE job (via OPTION PARSTD).

   Again, consider a VSE/POWER job containing 20 VSE jobs, each dealing with five job-specific disk files and thus requiring five DLBLs. If the first VSE job

[5] See “Label Area Segment (LAS)” on page 127 for details.

contained in the VSE/POWER job submits 100 DLBL statements to the partition's permanent LG, this may require seven LASs, whereas you need only one LAS to store each job's DLBLs in the partition's temporary LG.

Bottom line: If you have very specific label information required by only one or few jobs (or job steps) in a single partition, then submit it just in time to the partition's temporary LG. If you have label information common to many jobs in several partitions, submit it to the system or class LG rather than duplicating label information in the individual partition LGs.

What are free-usage labels for?

Each partition has a free-usage LG of its own, which will *not* be used by job control processing. That is, there are no job control statements to write LIRs into the free-usage LG.

It is up to third-party products to make use of it by means of the LABEL macro. The free-usage LG is important in terms of the label information search order, because it is ahead of the LGs administered by job control. As a matter of fact, the *z/VSE Guide to System Functions* manual conceals the existence of free-usage LGs in order to avoid confusion.

The true and complete label search sequence for GETLBL requests (as invoked by OPEN processing) is as follows:

- For static partitions and dynamic partitions with non-empty permanent partition LG:
  1. Free-usage partition LG
  2. Temporary partition LG
  3. Permanent partition LG
  4. System LG
- For dynamic partitions with empty permanent LG:
  1. Free-usage partition LG
  2. Temporary partition LG
  3. Class LG
  4. System LG

Is there a correlation between DLBL/TLBL statements and GETVIS consumption?

In case of dynamic partitions, there is a common misunderstanding about an apparent correlation between *dynamic space* GETVIS consumption on the one hand and TLBL/DLBL statements submitted on the other.

As we know by now, TLBL/DLBL statements produce LIRs, which are stored in LASs (2KB pieces of the data space). LASs are allocated to the respective LG. This allocation is reflected by a bitmap, which is either stored in a reserved pool of slots or in a 112-byte chunk of system GETVIS (LOC=ANY).

Thus there obviously is a correlation between the number of non-empty LGs and the consumption of system GETVIS (LOC=ANY). In the worst case, when 212 partitions are active using *all* three partition-related LGs (free-usage, temporary, and permanent), a little less than 63KB of system GETVIS (LOC=ANY) is required to store the bitmaps.

Are there any other storage requirements related to label processing?

1. In SVA 24, there is one 12KB chunk reserved for label processing. Furthermore, for each static partition, there is one 8KB chunk required by both job control and label processing. Together, these are 12 + 12*8 = 108KB of SVA-24 storage.
2. For each dynamic partition, this above-mentioned 8KB chunk is allocated in dynamic space GETVIS. It is always there, whether the dynamic partition submits DLBL/TLBL statements or not.
3. For each partition, label processing tracks GETLBL/GETNXL requests in the so-called history table. Place for 39 history table entries is provided in the 8KB chunk mentioned above.

   If a program needs to open more than 39 files, then the history table needs to be extended. Each 1104-byte extension is located in a GETVIS (SPACE=YES) subpool, the identifier of which starts with character string *IJBS*. It can be easily found in the output of GETVIS SVA,DETAIL.

How does /&-processing or program termination affect the lifetime of the different LGs?

There is a set of rules which needs to be considered when answering this question.

1. The OPTION statement with operands STDLABEL, CLASSTD, PARSTD or USRLABEL determines to which LG (think of it as a bucket) the subsequently submitted label information goes. Each OPTION (STDLABEL, CLASSTD, PARSTD or USRLABEL) immediately overwrites the preceding one. Program termination[6], /&- or JOB-processing will reset the OPTION to its default, which is USRLABEL.

   There are two differences between `OPTION xxx` on the one hand and `OPTION xxx=ADD` on the other:

   a. `OPTION xxx=ADD` only points to the bucket, where subsequent label information is to be stored. If the selected bucket is non-empty, all subsequent label information will be added to the existing one. Existing label information is preserved.

      `OPTION xxx` will first empty the corresponding bucket and then point to it. Existing label information is destroyed.

   b. `OPTION xxx` does not check whether subsequent DLBLs or TLBLs specify identical filenames.

      `OPTION xxx=ADD` checks for identical filenames and complains with message 1L30D (LABEL WITH SAME FILENAME IN SUBAREA) if necessary.
2. The buckets for free-usage labels remain empty, unless a third-party program decides to write label information into the partition's free usage LG. It is up to the third-party program to delete this label information or to empty the bucket.

   In z/VSE Job control statements will never touch free-usage labels. Therefore, they survive both program termination and /&-processing.

---

6 The term “program termination” (or “end of job step”) denotes the program swap, which takes place when an application program gives control back to the VSE operating system and a new copy of the job control program is loaded into the partition.

In z/VSE however there is one exception: dynamic partition free-usage buckets will be emptied during the processing of * $$ EOJ.

3. The buckets for partition permanent, class, and system labels survive both program termination and /&-processing. Dynamic partition permanent buckets will be emptied, however, during the processing of * $$ EOJ.

4. Partition temporary labels are cleared during /&- or JOB-processing. They are affected by program termination in the following way – when a new copy of the job control program is loaded into the partition, then diverse switches have their proper initial value. One of these switches controls what has to be done when a DLBL-EXTENT statement sequence or TLBL statement is to be processed.

   Initially this switch is off, saying “Clear the partition's temporary LG before processing the DLBL-EXTENT statement sequence or TLBL statement.” After processing the DLBL-EXTENT statement sequence or TLBL statement, the switch is turned on, saying “Don't clear the partition's temporary LG.”

   To illustrate, let's examine the following sample job

```
// JOB SAMPLE             1
// DLBL TEMP1             2
// OPTION PARSTD=ADD      3
// DLBL PERM              4
// EXEC LSERV JOBSTEP 1   5
// EXEC LSERV JOBSTEP 2   6
// DLBL TEMP2             7
// EXEC LSERV JOBSTEP 3   8
/&                        9
```

   Statement **1** clears the partition's temporary LG. There was no OPTION statement so far. The default option (USRLABEL) is in effect, therefore; and statement **2** submits label information for filename TEMP1 into the partition's temporary LG.

   Statement **3** overwrites the default (USRLABEL) in effect so far, and statement **4** submits label information for filename PERM into the partition's permanent LG.

   Statement **5** gives control to program LSERV. When the job control program is back in the partition, the above-mentioned switch is off; and any following DLBL or TLBL would cause the partition's temporary LG to be cleared. However, there is no DLBL/TLBL statement between the first job step and the second. Therefore, both LSERV outputs (statements **5** and **6**) contain label information for filename TEMP1.

   Since the preceding job steps caused the default option (USRLABEL) to be put in effect again and the switch is off, statement **7** first clears the partition's temporary LG (so TEMP1 is gone) and then submits label information for filename TEMP2 into the partition's temporary LG.

   The third job step (statement **8**) displays label information for TEMP2 (not for TEMP1), and the /&-statement finally clears the partition's temporary LG (statement **9**). Then TEMP2 is gone as well.

Notes:

a. If you submit job SAMPLE into the same partition for a second time, then the system issues message 1L30D when processing statement **4** .

b. If you execute a REXX procedure instead of a program (that is, you replace `EXEC LSERV` by `EXEC REXX=WHATEVER`), then you find the same behavior in terms of partition labels as before. This is because an EOJ macro is issued at the end of EXEC REXX processing, which causes a new copy of the job control program to be loaded into the partition.

How to Display the Label Request Trace Area

A wrong sequence of LABEL macro requests may lead to unpredictable results. Common request sequence errors cause message

```
1S40I SYSTEM ERROR, LABEL    RET. CODE=0C REASON CODE=xx
```

Such sequence problems are very intermittent, almost impossible to reproduce and difficult to analyze.

Phase $IJBSLA writes request information at the end of each partition's job control work area. COMREG field IJBJCWA points to this job control work area. The label request trace area starts at

```
COMREG.IJBJCWA + X'17F8'
```

The label request area starts with a fullword pointing to the next free entry. For each label request (that is each call of $IJBSLA) a 12-byte entry is written in wrap around mode. The layout is as follows:

```
2-byte    caller's task ID
2-byte    caller's function code (see macro LPLDCT for function codes)
4-byte    caller's register 1 (LPL address)
4-byte    caller's register 14 (return address)
```

These entries allow to check the last label requests, which caused the sequence error.

Symbolic Parameters

Scope of Symbolic Parameters

In the VSE/ESA releases up to VSE/ESA 2.3, the scope of symbolic parameters was restricted to a job: the set of symbolic parameters was released during end-of-job processing and had to be rebuilt (if applicable) by the next job. In this context *released* means, that GETVIS storage related to symbolic parameters (organized in subpools named IJPROCnnnn, where nnnn identifies the partition) is given back to the system (via FREEVIS).

In VSE/ESA 2.4 two additional sets of symbolic parameters were introduced. The first additional set contains parameters which are valid for the duration of a whole POWER job (or for the lifetime of a dynamic partition). The second one contains parameters valid for the whole system, which last until the system is IPLed again.

Thus we have three different parameter pools with different lifetimes:

1. (DOS) job parameters, which last until end-of-job (DOS EOJ) or until end-of-procedure, if defined in a cataloged procedure.

They are called *symbolic parameters at level n*, where n is the nesting level. n=0 if the parameter is defined in the job stream and 1≤n≤15 if the parameter is defined in a cataloged procedure.

There are independent parameter subpools for each of the 16 nesting levels and for each partition. Passing parameters (via EXEC PROC) means copying parameter entries from one subpool to another.

2. POWER job parameters, which last until * $$ EOJ processing (POWER EOJ).

   They are called *symbolic parameters at POWER job level*.

   There are independent POWER job parameter subpools for each partition.

3. System parameters, which last unit the system is shutdown or re-IPLed.

   They are called *symbolic parameters at system level*.

Symbolic parameters at POWER job or system level are defined exclusively by means of the SETPARM command. There is no difference, if you define them in a job or in a cataloged procedure.

Symbolic parameters at level n can be defined with the SETPARM, PROC or EXEC PROC statement. They can be passed from one nesting level to another.

Search Sequence

If a symbolic parameter is used in the operand field of a job control command or statement its value is retrieved according to following search sequence:

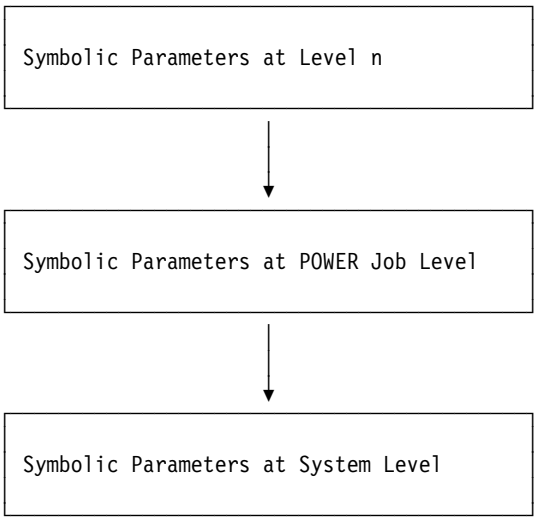


Figure 66. Search Sequence for Substitution of Symbolic Parameters

If the symbolic parameter is not found in any of the three parameter pools, message `1F3nD` is issued.

If, for example, the statement `// ASSGN SYS005,&CUU` is encountered on nesting level 4, then the system searches the following three pools for a substitution of `&CUU`:

1. symbolic parameters at level 4,
2. symbolic parameter at POWER job level
3. symbolic parameter at system level

If a user program (or JCL exit routine) resolves a symbolic parameter by means of macro GETSYMB, the same search sequence is in effect and a return code of X'10' in register 15 indicates, that the symbolic parameter was not found in any of the three parameter pools. If GETSYMB returns 0 in register 15 then the value returned in register 0 shows, in which parameters pool the parameter was found:

Reg0 = 1 found on level n, n is contained in COMREG.NSTLEVEL

Reg0 = 2 found on POWER job level

Reg0 = 3 found on system level

If a parameter name is specified in the condition expression of an IF statement, the same search sequence is in effect when retrieving this parameter's value.

Sample Scenario

The following sample scenario illustrates, how symbolic parameter values are retrieved according to their search sequence. For the scenario we make the following assumptions:

1. BG's JCL startup procedure ($0JCL.PROC) contains one SETPARM statement:

   ```
   // SETPARM SYSTEM,CUU1=120          1
   ```

2. The profile procedure of dynamic class Y (STDPROF.PROC) contains one SETPARM statement:

   ```
   // SETPARM PWRJOB,CUU1=130,CUU2=160    2
   ```

3. There is a cataloged procedure called ABC.PROC containing the following statements:

   ```
   // ASSGN SYS005,&CUU1
   /+
   ```

The following POWER job is processed in dynamic partition Y1:

```
* $$ JOB JNM=POWERJOB,CLASS=Y,DISP=D
// JOB DOSJOB1
// SETPARM CUU1=170,CUU3=181          3
// EXEC PROC=ABC
// ASSGN SYS005,&CUU1
// ASSGN SYS006,&CUU2
// ASSGN SYS007,&CUU3
/&
// JOB DOSJOB2
// ASSGN SYS005,&CUU1
// ASSGN SYS007,&CUU3
/&
* $$ EOJ
```

This job's output on SYSLST may look as follows:

```
// JOB DOSJOB1
// SETPARM CUU1=170,CUU3=181
// EXEC PROC=ABC
// ASSGN SYS005,130                    Substitution 1
EOP ABC
// ASSGN SYS005,170                    Substitution 2
// ASSGN SYS006,160                    Substitution 3
// ASSGN SYS007,181                    Substitution 4
EOJ DOSJOB1
// JOB DOSJOB2
// ASSGN SYS005,130                    Substitution 5
// ASSGN SYS007,&CUU3                  Substitution 6
1F34D  PARAMETER CUU3    NOT DEFINED
EOJ DOSJOB2
```

Explanation:

Substitution 1 It is nesting level 1 and &CUU1 is not defined as symbolic parameter at level 1. Next the system searches the pool of symbolic parameters at POWER job level. Because of **2** &CUU1 is substituted by 130. If cataloged procedure ABC had been invoked via `// EXEC PROC=ABC,CUU1` then &CUU1 would have been substituted by 170 (because of **3**).

Substitution 2 It is nesting level 0 and in **3** &CUU1 was defined as symbolic parameter at level 0. Thus &CUU1 becomes 170.

Substitution 3 It is nesting level 0 and &CUU2 is not defined as symbolic parameter at level 0. Next the system searches the pool of symbolic parameters at POWER job level. Because of **2** &CUU2 becomes 160.

Note that if POWERJOB had been submitted to a static partition &CUU2 would not have been found in any of the three pools, resulting in message 1F34D.

Substitution 4 It is nesting level 0 and in **3** &CUU3 was defined as symbolic parameter at level 0. Thus &CUU1 becomes 181.

Substitution 5 It is nesting level 0 and &CUU1 is not defined as symbolic parameter at level 0. Next the system searches the pool of symbolic parameters at POWER job level. Because of **2** &CUU1 becomes 130.

Note that if POWERJOB had been submitted to a static partition &CUU1 would have been substituted by 120 (because of **1**).

Substitution 6 It is nesting level 0 and &CUU3 is not defined as symbolic parameter at level 0. Next the system searches the pool of symbolic parameters at POWER job level, not found. Finally the system searches the pool of symbolic parameters at system level, not found. There is no substitution, therefore, and error message 1F34D is issued.

Storage and Capacity Considerations

For the parameter pools of symbolic parameter at POWER job or system level there are up to 10 2KB-chunks of GETVIS storage allocated to store symbolic parameter information records (SPIR's). This GETVIS storage is allocated with the attributes LOC=ANY and SPACE. Hence for a dynamic partition the required storage areas are taken from the dynamic space GETVIS area, whereas for static partitions (especially for symbolic parameters at system level) they are allocated in the system GETVIS area, preferably 31-bit. The GETVIS pool identifier is IJPROC, concatenated with a partition identifier. You can use the GETVIS command to display information related to system GETVIS storage allocated for symbolic parameters.

```
==> GETVIS SVA,DETAIL
    AR 0015 SUBPOOL       REQUEST  <--SVA-24-AREA---    --SVA-ANY-AREA-->
    AR 0015 IJPROC0050    SPACE                  8K                   4K
    AR 0015                         002C8000-002C9FFF    26085000-26085FFF
```

Figure 67. Sample Display of GETVIS Pools Related to Symbolic Parameters

Note that the GETVIS chunks below the 16MB line contain both control blocks for procedure handling and information records for symbolic parameters at level n. The ones above the 16MB line contain information records for symbolic parameters at POWER job level and symbolic parameters at system level.

It takes n+10 bytes to store a SPIR, where n denotes the length of the character string assigned to the parameter name. For n=50 you may store up to 329 symbolic parameters in each pool, for n=1 up to 1846 of them.

Consider the following scenario:

```
// SETPARM SYSTEM,PARM1='OLD VALUE'        1
...
// SETPARM SYSTEM,PARM1='NEW VALUE'        2
...
// SETPARM SYSTEM,PARM1='OTHER VALUE'      3
...
// SETPARM SYSTEM,PARM2='OTHERPARM'        4
```

Observe that the lengths of the character strings in **1**, **2** and **4** are equal to 9, whereas in **3** this length is 11. Let's assume that PARM1 has not been defined so far, so **1** creates a 19-byte SPIR for PARM1.

During processing of **2** the system recognizes, that PARM1 is already defined. Since the lengths of `OLD VALUE` and `NEW VALUE` are equal, the system does not create a second SPIR for PARM1, it just replaces the value information in the already existing SPIR.

During processing of **3** the system creates a second 21-byte SPIR for PARM1 and nullifies the first one.

In z/VSE 5.1 and before storage occupied by nullified SPIR's was no more available to store other SPIR's. In z/VSE 5.2 however nullified SPIR's with matching length are reused.

During processing of **4** the system recognizes, that the nullified SPIR for PARM1 has the right length to store the SPIR for PARM2.

So after completion of the four SETPARM statements we have two SPIR's:

- The 19-byte entry originally created for PARM1 and reused for PARM2
- The 21-byte entry for PARM1 created by **3**

This is reflected by the sequence of symbolic parameters displayed by the `QUERY SETPARM` command.

```
query setparm,system
AR 0015 IJBVMID = VSETEST4
AR 0015 PARM2   = OTHERPARM
AR 0015 PARM1   = OTHER VALUE
AR 0015 1I40I  READY
```

Figure 68. Sample Display of QUERY SETPARM,SYSTEM

Note that IJBVMID is an implicitely defined parameter containing the VSE guest's userid if running under z/VM.

Chapter 6. VSE/POWER

VSE/POWER Restart and Recovery after Abnormal End

Overview: After abnormal end or emergency termination, VSE/POWER warm start enters automatically spool file recovery indicated by messages 1QB7I and 1QB8I. Jobs found active (marked by DISP=*) will get their original disposition **D** or **K** and will execute again when a partition with a matching class becomes available. If VSE/POWER is terminated abnormally by a job this job might execute again after restart and will terminate VSE/POWER again. To prevent these jobs from starting again, use the VSE/POWER autostart statement **SET NORUN=YES**. This will set all active jobs found during recovery to temporary disposition **X** and will inform the operator.

Detailed Description:

To update the VSE/POWER Autostart statements, you need to find the PROC(s) executed to start the VSE/POWER partition. The steps below presume that VSE/POWER is running in partition **F1**.

- The SIR command shows which set of JCL procedures is used to start the z/VSE system. Search for output line with **JCL-PROC =** which names the JCL procedure, e.g. **JCL-PROC = $$JCLCT3**.
- The command **MAP F1** shows which VSE/POWER phase is active. E.g. **PHASE....: POWERCT3**.
- Use LIBR to LIST CPUVAR1.PROC from IJSYSRS.SYSLIB to lookup value of ENVNR.
- Issue command **PDISPLAY AUSTMT** on z/VSE console to list all executed AUTOSTART statements.

```
pdisplay austmt
AR 0015 1C39I COMMAND PASSED TO VSE/POWER
F1 0001 1R48I  001 ***  BEGIN OF AUTOSTART INFORMATION  ***
F1 0001 1R48I  002 SET SJECL=YES
F1 0001 1R48I  003 SET CONFIRM=PRELEASE,QUEUE,ALL
F1 0001 1R48I  004 SET OUTEXIT=SAS
F1 0001 1R48I  005 DEFINE L,CICSDATA,3F00,1,255,*
F1 0001 1R48I  006 FORMAT=NO
F1 0001 1R48I  007 PSTART BG,A0I
F1 0001 1R48I  008 READER=FEC
 ...
```

Now search within e.g. $1JCLCT3 how POWERCT3 phase is executed

1. EXEC POWERCT3 (direct call to POWER start phase)
2. EXEC PROC=PWSTRCT3 (call to nested procedure)
3. EXEC PROC=POWSTRT&XENVNR,... (call to nested proc., IBM default)

Insert new **SET NORUN=YES** after **each location** of SET OUTEXIT=SAS in either

1. $1JCLCT3 procedure
2. PWSTRCT3 procedure
3. POWSTRTA or POWSTRTB or POWSTRTC procedure. Suffix A, b or C was defined during initial installation and can be obtained from CPUVAR1.PROC

After changing the VSE/POWER startup, verify your changes by performing a re-IPL. PDISPLAY AUSTMT should now show SET NORUN=IGN in line 005.

```
pdisplay austmt
AR 0015 1C39I COMMAND PASSED TO VSE/POWER
F1 0001 1R48I  001 ***  BEGIN OF AUTOSTART INFORMATION  ***
F1 0001 1R48I  002 SET SJECL=YES
F1 0001 1R48I  003 SET CONFIRM=PRELEASE,QUEUE,ALL
F1 0001 1R48I  004 SET OUTEXIT=SAS
F1 0001 1R48I  005 SET NORUN=IGN
F1 0001 1R48I  006 DEFINE L,CICSDATA,3F00,1,255,*
F1 0001 1R48I  007 FORMAT=NO
F1 0001 1R48I  008 PSTART BG,A0I
F1 0001 1R48I  009 READER=FEC
 ...
```

VSE/POWER queue file recovery will now assign temporary disposition **X** to all active jobs. This includes jobs for subsystems CICS, VTAM, TCPIP,... . To avoid that jobs starting subsystems receive disposition X, add parameter **NORUN=IGN** on the *** $$ JOB** card of each job starting a subsystem. Submit the new job(s) and delete the old job(s) from the RDR queue. In addition you may adapt all jobs located in IJSYSRS.SYSLIB which start subsystems and are loaded during VSE/POWER Cold start by procedure COLDJOBS.PROC or your own procedure called during VSE/POWER cold start.

If you are unable to include the **SET NORUN=YES** statement in your VSE/POWER autostart you can trigger this function by setting the first bit of the UPSI byte to '1' before the VSE/POWER phase is executed. Issue the AR command 'PAUSE f1' for the VSE/POWER partition before VSE/POWER is started. The partition will then wait for the operator response after the first '// JOB' or '/&' or '// EXEC phase' statement read. Note that a '// JOB' statement should preceed the execution of the VSE/POWER phase. Now you can enter '// UPSI 1' on the console and all jobs found active during recovery will get the new disposition 'X' which prevents their execution.

VSE/POWER Handling Spool Space Shortage

Overview: VSE/POWER jobs and outputs are stored in RDR, LST, PUN and XMT queues. Each queue entry consist of a queue record and one or more DBLK groups. Output duplicates are an exception to this rule and consist of a queue record only. The size of the queue file EXTENT (1.0) determines the number of queue records (1.1) and the number and size of data file EXTENTs (2.0) divided by the DBLK size (2.6) defines the number of DBLKs. The number of DBLKs divided by the DBLK group size (2.7) then defines the number of DBLK groups (2.1). For each new job or output to be created by VSE/POWER, a queue record and at least one DBLK group are required. If all queue records are already in use, a task creating a new job or output will issue message

```
1QF4A      NO FREE QUEUE RECORD AVAILABLE FOR task,cuu
```

If no DBLK group is available to start or to continue spooling, message

```
1Q38A NO DASD SPACE AVAILABLE FOR TASK, CUU
```

will be issued. In both cases the VSE/POWER task is put into wait state until a free queue record respectively a free DBLK group becomes available. As long as

the VSE/POWER task waits for the missing resource, also the spooling partition will wait for IO completion. To avoid such situations, the following guide lines may help.

Causal research: Whenever messages 1QF4A or 1Q38A are issued, issue the PDISPLAY STATUS command to check, whether all queue records (1.3) or DBLK groups (2.3) are in use. Note, that 10 queue records and 20 DBLK groups are reserved as cushion for critical tasks.

```
     1R46I  QUEUE FILE     IJQFILE
(1.0)   TOTAL NUMBER OF TRACKS                              15 TRACKS
(1.1)   TOTAL NUMBER OF QUEUE RECORDS                     1886 RECORDS
(1.2)    FREE QUEUE RECORDS (INCL. 10 FOR CUSHION)        1630 RECORDS
(1.3)    USED QUEUE RECORDS                                256 RECORDS
(1.4)     QUEUE RECORDS IN DELETION                          0 RECORDS
(1.5)     QUEUE RECORDS LOST DUE TO I/O OR LOGIC ERROR       0 RECORDS
        MAX. NO. OF Q-REC'S USED IN PRESENT SESSION         257 RECORDS
        MAX. NO. OF Q-REC'S USED SINCE LAST COLDSTART       470 RECORDS
        QUEUE FILE STOR. COPY IN PART. GETVIS (TOTAL)       708 K-BYTES
        QUEUE FILE STOR. COPY, PART IN PART. GETVIS-31      708 K-BYTES
     1R46I  DATA FILE      IJDFILE                             1 EXTENT
(2.0)   TOTAL NUMBER OF TRACKS                            1920 TRACKS
(2.1)   TOTAL NUMBER OF DBLK-GROUPS                       1680 GROUPS
(2.2)    FREE DBLK-GROUPS (INCL. 20 FOR CUSHION)          1354 GROUPS
(2.3)    USED DBLK-GROUPS                                  326 GROUPS
(2.4)     DBLK-GROUPS IN DELETION                            0 GROUPS
(2.5)     DBLK-GROUPS LOST DUE TO I/O OR LOGIC ERROR         0 GROUPS
        MAX. NO. OF DBLK-GPS USED IN PRESENT SESSION        328 GROUPS
        MAX. NO. OF DBLK-GPS USED SINCE LAST COLDSTART      328 GROUPS
(2.6)   DATA BLOCK GROUP SIZE                                 8 DBLKS
(2.7)   DATA BLOCK SIZE                                   7548 BYTES
        SPOOL LIMIT PERCENTAGE                             90 %
```

If the output from PDISPLAY STATUS shows that a lot of queue records / DBLK groups have been lost due to I/O or logic errors (1.5/2.5), search for error messages in the hardcopy file and whether a dump was taken.

- 1QF8I nnnnnnnnn FREE DBLK GROUP(S) OF A SUBCHAIN‘ (ABOUT mmm%) LOST
- 1QFAA USED DBLK GROUP FOUND IN A FREE DBLK GROUP SUBCHAIN
- 1QFBA FREE DBLK GROUP FOUND IN RETURNED QUEUE ENTRY
- 1QFCA MISMATCH OF GROUP COUNT AND ACTUAL NUMBER OF DBLK GROUPS
- 1QFDA MISMATCH OF SUBCHAIN COUNT AND ACTUAL NUMBER OF FREE GROUPS

Verify that queue file and data file were neither copied to the failing system while VSE/POWER was up and running nor accessed by more than one VSE/POWER system in parallel and report the problem to IBM.

First remedies: If the statistics show that neither queue records nor DBLK groups are lost but queue records are short, search for queue entries which are no longer needed and can be deleted. Use the PDISPLAY LST|PUN|XMT with SORT=OLD to search for oldest outputs. The amount of entries to be displayed is 16, for more entries specify LIMIT=nn.

```
1R46I  FOR 'D LST,.,LIMIT=016,SORT=OLD' COLLECTED 016 OF 00217 ENTRIES
1R46I    LIST QUEUE   P D C S  PAGES  CC FORM B
1R46I  VTAM511  00006 3 D A        6   1       D=12/08/2011 T=10:12:20
1R46I  TCPIP511 00007 3 D A       10   1       D=12/08/2011 T=10:12:23
1R46I  CICS511  00002 3 D A       39   1       D=12/08/2011 T=10:12:23
1R46I  VTAM511  00025 3 D A        6   1       D=12/08/2011 T=11:28:28
1R46I  TCPIP511 00027 3 D A       11   1       D=12/08/2011 T=11:28:31
1R46I  CICS511  00026 3 D A       39   1       D=12/08/2011 T=11:28:31
1R46I  VTAM511  00034 3 D A        4   1       D=12/08/2011 T=11:34:58
1R46I  TCPIP511 00036 3 D A       10   1       D=12/08/2011 T=11:35:01
1R46I  CICS511  00035 3 D A       39   1       D=12/08/2011 T=11:35:01
1R46I  IESUPDCF 00038 3 D A        2   1       D=12/08/2011 T=11:36:04
1R46I  VTAM511  00040 3 D A        7   1       D=12/08/2011 T=11:37:08
1R46I  CICS511  00041 3 D A       40   1       D=12/08/2011 T=11:37:19
1R46I  CATJCL51 00042 3 D A 3      4   1       D=12/08/2011 T=11:39:01
1R46I  TCPIP511 00043 3 D A       33   1       D=12/08/2011 T=11:39:09
1R46I  TCPINOSA 00304 3 K I        3   1       D=12/08/2011 T=13:49:06
1R46I  UPDAYSCF 02443 3 D A        5   1       D=12/09/2011 T=07:38:13
```

To determine the size of a queue entry in DBLK groups, use PDISPLAY command with ,FULL=YES, e.g. PDISPLAY LST,IESUPDCF,38,FULL=YES. If no disposable queue entries can be found, perform a POFFLOAD PICKUP and save queue entries to tape before deleting them. The same commands & methods should be used when DBLK groups are short. In addition, the PDISPLAY BIGGEST command will show the 16 largest queue entries in all queues including CRE and DEL queue.

```
1R4BI 016 BIGGEST SORTED C I CARD/LINE    DBGP  QNUM SUF     PAGES QUE
1R4BI 001 PAUSEBG  47833 A L     68260 0000146 01658          1558 CRE
1R4BI 002 TCPHUNDT 02687 H L      7510 0000014 01868           251 LST
1R4BI 003 TCPIP511 02692 A L      6766 0000013 01854           104 LST
1R4BI 004 TCPIP511 26508 A L      3298 0000006 01690            51 LST
1R4BI 005 CICS511  02516 A L      2379 0000005 01871            44 LST
1R4BI 006 CICS511  00002 A L      2145 0000004 00023            39 LST
1R4BI 007 CICS511  00026 A L      2113 0000004 00026            39 LST
1R4BI 008 CICS511  58137 A L      1794 0000004 01668            31 CRE
1R4BI 009 CICS511  53042 A L      2275 0000004 01672            43 LST
1R4BI 010 CICS511  49472 A L      2274 0000004 01675            42 LST
1R4BI 011 CICS511  49433 A L      2266 0000004 01678            42 LST
1R4BI 012 CICS511  49413 A L      2271 0000004 01680            42 LST
1R4BI 013 CICS511  38261 A L      2123 0000004 01688            39 LST
1R4BI 014 CICS511  26507 A L      2142 0000004 01691            39 LST
1R4BI 015 CICS511  29254 A L      2346 0000004 01694            44 LST
1R4BI 016 IPWSEGM  41909 A L      2151 0000004 01697            90 LST
```

If you find large queue entries in creation, use the Interactive Interface to analyze the output, whether e.g. a dump or trace is taken or whether the job seems to produce output in a loop. You can cancel such a job by the VSE/POWER command PFLUSH partition-id. One DBLK group from the cushion will be enabled for the output in creation, thus allowing completion of the job. When the job has completed, delete the output to free the DBLK groups again (POFFLOAD PICKUP the output in case it is needed for analysis). If analysis of the output revealed a dump which should be preserved, use PSEGMENT partition-id,cuu to segment the output and offload the first part to tape (or print it to Z/VM or get it via FTP,...). Also PSEGMENT enables one DBLK group from the cushion to continue spooling until the next page break forces segmentation.

Long term solution: Analyze the queue and data file usage.

- Is expiration management used for outputs?
- Are outputs regularly offloaded, e.g. by POFFLOAD PICKUP or SAVE?
- Are offloaded outputs deleted thereafter?
- Did the workload increase?
- Are the VSE/POWER queues used as a repository?

If the answers to above questions indicate that queue and data file should be extended, keep the balance between both files in mind. Whenever queue or data file is increased, make sure that the size of the other file does still match. The PDISPLAY Q command output provides a brief overview of how many queue records and DBLK groups are currently used. The ratio of used DBLK groups and queue records for your system will be greater than 1 (unless a lot of duplicates are used). A similar or slightly larger ratio should be kept for total DBLK groups and queue records after expanding the queue file and / or data file.

To increase the queue file or data file a cold start is no longer necessary. You may increase either the queue file or the data file during a VSE/POWER warm start. For details see the VSE/POWER Administration & Operation manual, chapter 1.

VSE/POWER Storage Management

Summary: VSE/POWER storage management handles PFIXable storage and GETVIS, both residing within the VSE/POWER partition. Storage is claimed for and used by VSE/POWER internal tasks and released when a task ends. Therefore the amount of used storage depends on the number of active tasks and is limited by the partition layout. Each VSE/POWER internal task requires at least 768 byte PFIXED storage for the VSE/POWER Task Control Block and a DBLK work area in partition Getvis below 16MB.

Detailed Description:

The partition layout is defined by an ALLOC and a SIZE specification typically issued by ALLOC.PROC executed in $0JCL.PROC. The SETPFIX LIMIT is specified in $1JCL.PROC. This divides the partition into 3 sections:

- The partition starts with the SETPFIX LIMIT storage, where VSE/POWER places TCBs, IO areas, control blocks and other work areas which are all PFIXed to avoid page faults. The SETPFIX LIMIT is shown in the **PDISPLAY STATUS** report in line (1) and the maximum and current usage are shown in lines (1.1) and (1.2).
- The next area is the pageable area into which VSE/POWER loads its executable code during initialization. The size of this area is equal to the specified SIZE minus SETPFIX LIMIT and is the sum of lines (2.1) and (2.2).
- The difference between ALLOC and SIZE builds the partition GETVIS where VSE/POWER places other work areas in GETVIS-24 and the copy of the queue file, which may reside in GETVIS-31 if available. GETVIS-24 and its usage is shown in lines (3) to (3.3).

Here is an excerpt from a **PDISPLAY STATUS** report:

```
     1R46I  GENERAL STORAGE STATISTICS
(1)     FIXABLE STORAGE ALLOCATED TO VSE/POWER                200 K-BYTES
(1.1)    MAX. NO. OF K-BYTES FIXED IN PRESENT SESSION          88 K-BYTES
(1.2)    CURR. NO. OF K-BYTES FIXED IN PRESENT SESSION         84 K-BYTES
(1.3)    NO. OF TIMES TASKS WAITING FOR PFIXED STORAGE          0 TIMES
(2.1)   VIRTUAL STORAGE OCCUPIED BY VSE/POWER PHASES          839 K-BYTES
(2.2)   UNUSED STORAGE REMAINING BELOW SIZE BOUNDARY           22 K-BYTES
(3)     TOTAL PART. GETVIS-24 STORAGE ALLOCATED             10205 K-BYTES
(3.1)    MAX. GETVIS-24 REQUESTED IN PRESENT SESSION         106 K-BYTES
(3.2)    CURRENT GETVIS-24 AMOUNT REQUESTED                   80 K-BYTES
(3.3)    NO. OF TIMES TASKS WAITING FOR GETVIS-24 STOR         0 TIMES
        SYSTEM GETVIS STORAGE USED BY VSE/POWER               36 K-BYTES
```

Also the number of active tasks are shown in the **PDISPLAY STATUS** report:

```
     1R46I  GENERAL TASK STATISTICS
        MAX. NO. OF TASKS ACTIVE AT ONE POINT IN TIME         20 TASKS
         CURRENT NUMBER OF ACTIVE TASKS                       19 TASKS
```

The VSE/POWER storage statistics should be checked regularly to avoid the following problems:

- Message 1Q05I PAGEABLE AREA nnnK TOO SMALL is issued during VSE/POWER start and either VSE/POWER is terminated or PNET function is not initialized or VSE/POWER user exits are not loaded.

  Increase the SIZE (and if appropriate the ALLOC) of the F1 partition in ALLOC.PROC. Monitor line (2.2) to keep approximately 40K for future growth of VSE/POWER phases.

- Message 1Q59I task, cuu WAITING FOR REAL/PFIXED STORAGED is issued and VSE/POWER tasks are forced to wait that other VSE/POWER tasks end and release PFIXED storage.

  Check line (3.3) whether this problem happened more than once and if so, search the console log for message 1Q59I. Which tasks did request PFIXED storage and how much storage was requested? Analyse the number of active tasks and if the SETPFIX LIMIT is too small, increase it in $1JCL.PROC by 50K or 100K. Increase also the SIZE and ALLOC of the F1 partition in ALLOC.PROC accordingly. After the next IPL check line (1.1) for 20K SETPFIX LIMIT still available.

- Message 1Q85I task, cuu WAITING FOR GETVIS-24 STORAGE, xxx BYTES is issued and VSE/POWER tasks are forced to wait that other VSE/POWER tasks end and release GETVIS-24.

  Check line (3.3) whether this problem happened more than once and search console log for additional messages 1Q85I. Analyse the messages and increase the ALLOC of the F1 partition in ALLOC.PROC. Verify in **PDISPLAY STATUS** report that the queue file copy already resides in partition GETVIS-31.

When messages 1Q59I and 1Q85I are issued again although the storage areas have been increased, you should check for misuse. VSE/POWER user exits may request inappropriate amounts of GETVIS. **PDISPLAY EXIT** will show whether and which VSE/POWER user exits are active:

```
1R4AI  EXITTYPE STATE    NAME     WA-SIZE ADDRESS  EXITSIZE WU
1R4AI  JOBEXIT  ENABLED  RDREX1   00050   001F0200 01234    NP
1R4AI  OUTEXIT  ENABLED  PRTEX5   01000   001E0000 00910    PA
1R4AI  NETEXIT  ENABLED  NETEXIT1 00040   00100200 00502    NP
1R4AI  XMTEXIT DISABLED  XMTEX1   00112   00120000 00300    PA
```

Each VSE/POWER task is equipped with its own work area when using an exit. In the example above the OUTEXIT will need 4K GETVIS for each printer and for each Spool Access Support task accessing the LST & PUN queue, e.g. for CICS/RCF, TCP/IP FTP, LPR or EMAIL. User exits may request PFIXED storage or GETVIS directly, bypassing VSE/POWER storage management. Then the VSE/POWER storage statistics are no longer in sync with the output of the GETVIS command. Use **GETVIS F1,DETAIL** to receive a detailed report about partition GETVIS usage. GETVIS requested and used by VSE/POWER will belong to a GETVIS subpool with a name starting with **IPW**.

Another cause of excessive usage of PFIXED storage or GETVIS may be a VSE/POWER Spool Access Support application generating tasks in a loop. For an overview which VSE/POWER tasks are currently active use **PDISPLAY TASKS**

```
1R48I  ***   BEGIN OF DISPLAYING VSE/POWER TCB'S       ***
1R48I  TID ,CUU,TCBADR,T,PHASE(ADDR),REG12 ,STATE(DETAIL)
1R48I  YTES,   ,50A000, ,TV (5C9F00),5C9FE0,C(R01=50A034)
1R48I  JSPM,   ,50C840, ,SM (5CA300),5CA50C,M(R01=2F20B8)
1R48I  O CP,   ,501340, ,CM (53B000),53B010,R(----------) (D TASKS)
1R48I  XMAS,   ,50BAC0, ,XM (5BCB00),5BCE26,M(R01=50BC58)
1R48I  XSAS,PSP,512900, ,XTS(5C3500),5C35E0,M(R01=512A98) (00004)
1R48I  XSAS,   ,512B00, ,XTS(5C3500),5C35E0,M(R01=512C98) (00015)
1R48I  XSAS,   ,512E00, ,XTS(5C3500),5C35E0,M(R01=512F98) (00016)
1R48I  XSAS,   ,513100, ,XTS(5C3500),5C35E0,M(R01=513298) (00017)
1R48I  XSAS,   ,513400, ,XTS(5C3500),5C35E0,M(R01=513598) (00018)
1R48I  XSAS,   ,513700, ,XTS(5C3500),5C35E0,M(R01=513898) (00019)
1R48I  NTFY,   ,512600, ,NS (5C5C00),5C6124,M(R01=512828)
1R48I  DPST,   ,50A580, ,DP (59B900),59B9FA,X(R01=50A320)
1R48I  E BG,FEC,50D000, ,XRE(568A00),569998,C(R01=50D034)
1R48I  E F2,FEC,50D300, ,XRE(568A00),569998,C(R01=50D334)
1R48I  E F3,FEC,50CD00, ,XRE(568A00),569998,C(R01=50CD34)
1R48I  E F4,FEC,50D9C0, ,XRE(568A00),569DA8,Q(R01=50D9E0)
1R48I  E F5,FEC,50E000, ,XRE(568A00),569DA8,Q(R01=50E020)
 . . .
```

In the above example 6 SAS tasks 00004, 00015 - 00019 are shown, to receive detailed information, issue "PDISPLAY A,SAS" :

```
1R48I  SAS,00004, SAS=SYSCICS5,WALB, REQ=PUT, JSTRAN1 ,02294,A
 864 RECORDS SPOOLED
1R48I  SAS,00015, SAS=SYSCICS5,
1R48I  SAS,00016, SAS=SYSCICS5,
1R48I  SAS,00017, SAS=SYSCICS5,
1R48I  SAS,00018, SAS=SYSCICS5,
1R48I  SAS,00019, SAS=SYSCICS5,
```

SAS task 00004 is creating a job or an output. SAS tasks 00015 - 00019 were created one after the other and do not execute any function. If you are in urgent need for storage you may cancel such SAS tasks by VSE/POWER command "PSTOP SAS,connect-id", e.g. "PSTOP SAS,00015". To prevent that an application program creates too many SAS tasks, VSE/POWER defines a limit of 250 SAS tasks active in parallel. The limit can be modified by command "PVARY MAXSAS,nnnnnn".

Schedule VSE/POWER-Job More Than Once Per Day

Summary: VSE/POWER supports a new function concerning Time Event Scheduling. Usually, dispatchable jobs run just once. Using time event scheduling parameters, a job may run more than once; for example, every first day of a month, every Monday, or even daily. Currently, a job can run only once on a specific scheduled day.

Using the new operand DUEFRQ (Due Time Frequency) in the VSE/POWER * $$ JOB statement, a job can be scheduled to run more than once on a specific day.

Detailed Description:

DUEFRQ Operand In * $$ JOB Statement The VSE/POWER * $$ JOB statement may now contain an operand which specifies a time after which a job is to be rescheduled and an ending time after which no more scheduling should occur.

```
DUEFRQ=(iimm,llnn)
```

The Due Time Frequency operand specifies an interval using ii for hours and mm for minutes (from 0001 through 2359) after which the job has to be rescheduled and specifies a last time using ll for the hour and nn for the minute (from 0000 through 2400) after which the job is no longer to be scheduled for processing. Leading zeros must be specified.
This operand requires a DUETIME to be specified which defines the time when scheduling has to occur for the first time.
This operand is accepted only if DAILY or a Weekday-List has been specified for the operand DUEDAY. DUEFRQ is rejected, if a day-list has been specified for DUEDAY.
RERUN=YES must not be used for jobs with a DUEFRQ specification. Instead RERUN=NO becomes the default. Hence, if a job with a DUEFRQ specification misses a scheduling event due to system down time, the job gets scheduled at the next scheduling event.

Following samples describe the usage of the operands:

```
DUETIME=0000,DUEDAY=DAILY,DUEFRQ=(0030,2400)
```

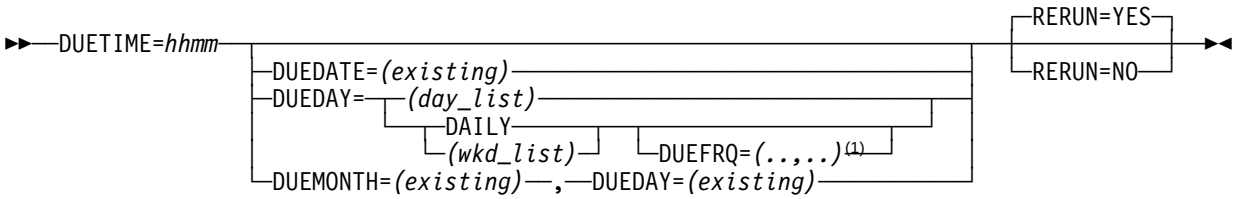
A job with the above operands is scheduled every 30 minutes every day. If the job is read in at 3:01 p.m., the job is scheduled for the first run at 3:30 p.m.

```
DUETIME=0700,DUEDAY=(MON-FRI),DUEFRQ=(0100,1700)
```

A job with the above operands is scheduled every hour between 7 a.m. and 5 p.m. on every day from Monday through Friday (including any holidays). The first time the job is scheduled is at 7 a.m. and the last time is at 5 p.m.
If a job with the above operands is scheduled e.g. for the next run at 9 a.m. and its processing is finished after 10 a.m., the job gets rescheduled for 11 a.m. The scheduling at 10 am is missed. This behavior can not be changed by the RERUN operand.

Note: A job is scheduled every n minutes regardless how long the job needs for processing. If for example a job is scheduled every 5 minutes and is scheduled for 7:00 for the next run and finishes its processing at 7:04, the job is rescheduled for 7:05 and not for 7:09. If however the job finishes its processing at 7:06, the job is rescheduled for 7:10. As no seconds are considered when calculating the scheduling time, it even may occur that a job finishes its processing at 7:04:59 and gets rescheduled for 7:05:00.

The syntax diagram for the * $$ JOB statement is updated (see Figure 69 on page 149) with the following part including the operand DUEFRQ.



Note:
[1] For DUEFRQ do not specify RERUN=YES, instead RERUN=NO becomes the system default.

*Figure 69. * $$ JOB statement - syntax updates including the operand DUEFRQ*

Displaying The Information Of The DUEFRQ Operand: Once a job has been read in by VSE/POWER, the specified values for the DUEFRQ operand can be displayed by using the VSE/POWER PDISPLAY command together with the FULL=YES operand. The next scheduling time and date is displayed in the first line after the constant RUN=. The time in the second line after the constant DUETIME= is the time specified for DUETIME and identifies the time when the job has to be scheduled the first time for a selected day.

To demonstrate the displayed scheduling information assume the following sample jobs:

```
* $$ JOB JNM=HUFQ001,DUETIME=0000,DUEDAY=DAILY,DUEFRQ=(0030,2400)

* $$ JOB JNM=HUFQ002,DUETIME=0700,DUEDAY=(MON-FRI),DUEFRQ=(0100,1700)

* $$ JOB JNM=HUFQ003,DUETIME=0600,DUEDAY=FRI,DUEFRQ=(0105,1300)
```

And assume they were read in on Tuesday, August 10th, 1999, at 9:15 a.m. and the subsequent display been requested immediately:

```
PDISPLAY RDR,HU*,FULL=YES

1R46I  READER QUEUE   P D C S  CARDS
1R46I  HUFQ001  01092 3 K A        3 RUN=09:30,08/10
       D=08/10/1999 DBGP=000001
       DUETIME=00:00 DUEDAY=DAILY DUEFRQ=(00:30,24:00) RERUN=NO
       QNUM=00469
1R46I  HUFQ002  01093 3 K A        3 RUN=10:00,08/10
       D=08/10/1999 DBGP=000001
       DUETIME=07:00 DUEDAY=(MON,TUE,WED,THU,FRI) DUEFRQ=(01:00,
                                                    17:00) RERUN=NO
       QNUM=00490
1R46I  HUFQ003  01094 3 K A        3 RUN=06:00,08/13
       D=08/10/1999 DBGP=000001
       DUETIME=06:00 DUEDAY=(FRI) DUEFRQ=(01:05,13:00) RERUN=NO
       QNUM=00491

PDISPLAY WRUN

1R46I  READER QUEUE   P D C S  CARDS (WAIT FOR RUN SUBQUEUE)
1R46I  HUFQ001  01092 3 K A        3 RUN=09:30,08/10
1R46I  HUFQ002  01093 3 K A        3 RUN=10:00,08/10
1R46I  HUFQ003  01094 3 K A        3 RUN=06:00,08/13
```

Altering The Information Of The DUEFRQ Operand: Once a job has been read in by VSE/POWER, the specified values for the DUEFRQ operand can be nullified by using the VSE/POWER PALTER command together with the DUETIME=NULL operand. As till now, all specifications of time event scheduling time are ignored and lost.

Transferring Jobs With DUEFRQ Information To Other Systems: If a job gets read in by VSE/POWER, the DUEFRQ operand is processed and the specified values are saved for this job. If the job gets transferred to any other VSE/POWER system, this information never gets lost. The transfer may for example happen via PNET or by offloading the job to a tape and reloading it somewhere else. Whenever the job enters the reader queue the next scheduling date is calculated anew. Whenever the job is passed from one transmission queue to another transmission queue (at a store and forward node), no next scheduling date is calculated.

DUEFRQ Information On Backlevel Systems Without DUEFRQ Support: Once a job has been read in by a VSE/POWER system, the job may get transferred to another VSE/POWER system, which might be at a different level. If a job with DUEFRQ information is sent to a system without DUEFRQ support, the following is true:

1. No information about DUEFRQ is displayed when using the FULL=YES operand.
2. The next scheduling time is the value been determined as the next processing time at the originating system. The next scheduling day is calculated according to the usual rules when entering a system. (Note, the next scheduling time is the time displayed after the constant RUN= when using the PDISPLAY command.)
3. The time displayed for DUETIME is the time for the next scheduling event. This time might now differ on the receiving system from that time which was originally specified for the DUETIME operand on the transmitting system.

4. If at the sending system a job enters the transmit queue through the read-in process and is transmitted, then at the receiving system the next scheduling time is the time specified for the DUETIME operand.
5. If a job gets transferred from a system with DUEFRQ support to a system without DUEFRQ support and gets transferred back to the originating or another system with DUEFRQ support, all information of the DUEFRQ operand is available to the receiving system with DUEFRQ support.

Information Of The DUEFRQ Operand On Shared Systems: It is recommended to have all systems of a shared environment at the same service level. If however one system has included the DUEFRQ support, but the other has not, the following happens. If a job with DUEFRQ information gets processed by the system without the DUEFRQ support, the job does not get rescheduled once more for the same day. But the next scheduling date is calculated according to the next scheduling time for another day. If, next time, the job gets processed by the system with DUEFRQ support, the job may get rescheduled again for another run on the same day, as the DUEFRQ information is still available.

Interface To Application Programs: There exists also a fixed format display for programs using the spool-access support. This format is not changed or extended in order to display the information about the DUEFRQ operand. One reason for this is, that already today no day-list nor a month-list are contained within this format.

Command Driven Job Output Segmentation

solving Datafile-Full Condition

Summary: When the datafile is full and jobs continue to spool output, these jobs stall, until DBLK groups are made available, for example by deleting VSE/POWER queue entries. In previous versions of VSE/POWER there were two problems:

- if the job could not be flushed, there was no way to segment the output dynamically by command to process the created segment, delete it afterwards and free disk space for further processing and allow the job to continue.
- even when the job was flushed by 'PFLUSH partition' command, it still stalled until at least one DBLK group becomes available, before flushing is processed (prematurely end the output with message 1Q39I, add output entry to e.g. LST queue).

For the purpose of immediate segmentation, a new command PSEGMENT has been provided beginning with VSE/POWER 6.6 - see VSE/POWER Administration & Operation.

Internal Queue Entry Number Used in Displays and Commands

Summary: The internal queue entry number, which is for example needed for SAS Direct GET, was not supplied by VSE/POWER in queue displays on the console, but since the APARs listed below, it is shown in all queue displays issued with option 'FULL=YES'.

This queue entry number can now be used as an additional search argument in all queue manipulation commands (PALTER, PDELETE, PHOLD, PRELEASE), to select a specific queue entry out of a set of queue entries with same attributes.

If a job with disposition 'L' is for example not released, but altered to disposition 'K' multiple times, it will create output with the same jobnumber, each time it is started. The result is a set of identical queue entries, which can only be distinguished by their internal queue entry number as now shown in a queue display with 'FULL=YES'.

Detailed Description

VSE/POWER shows now the internal queue entry number in the return of a 'PDISPLAY queue,FULL=YES'.

Example:

```
d rdr,WST*,full=yes
1R46I  READER QUEUE   P D C S  CARDS
1R46I  WST07    00027 5 D A       10 FROM=(WALB)
       D=12/09/1996 DBGP=000001
       SECN=ZONE04 QNUM=01234
```

QNUM
shows the internal queue entry number in decimal format.

The new operand 'CQNUM=nnnnn' is added to the list of **keyword search operands** for PALTER, PDELETE, PHOLD and PRELEASE commands and specifies the Current Queue entry NUMber as presented by 'PDISPLAY queue,...,FULL=YES'

CQNUM=nnnnn
specifies the unique queue entry number assigned to the job by VSE/POWER. 1 to 5 digits may be specified. You can use the PDISPLAY command with option 'FULL=YES' to obtain this number.

Note: When this operand is specified, only 1 job can match the search criteria.

How to use CQNUM: Assuming, that 'PDISPLAY queue,FULL=YES' shows identical entries as in the following example.

```
d lst,pausebg
AR 0015 1C39I COMMAND PASSED TO VSE/POWER
1R46I    LIST QUEUE   P D C S  PAGES  CC FORM
1R46I  PAUSEBG  00027 3 D A       10   1 TO=(SYSA) FROM=(SYSA)
       D=12/01/2000 DBGP=000002 L=00000426
       QNUM=01234
1R46I  PAUSEBG  00027 3 D A        8   1 TO=(SYSA) FROM=(SYSA)
       D=11/01/2000 DBGP=000001 L=00000379
       QNUM=05517
1R46I  PAUSEBG  00027 3 D A       49   1 TO=(SYSA) FROM=(SYSA)
       D=11/01/2000 DBGP=000007 L=00002168
       QNUM=03490
```

Then a queue manipulation command would normally address either the first queue entry or all queue entries.

Use the new operand 'CQNUM', to select only a single queue entry:

```
PALTER LST,PAUSEBG,...,CQNUM=5517,USER=MARTIN
```

Now only the queue entry with the unique queue entry number '5517' will be altered, if all other specified search operands match also.

Transmission Queue Disposition

Summary: When jobs or output enter the transmit queue their default disposition is 'D'. After transmission the job/output becomes deleted. For multiple transmissions the disposition must be 'K' or 'L' which is difficult or even impossible to achieve.

Detailed Description

By default, jobs and output entries have obtained transmission disposition D, when they entered the XMT queue. Now the transmission disposition D|H|K|L can be specified by the new TDISP=D|disposition operand for jobs in the * $$ JOB statement and for output entries in the * $$ LST/PUN statement.

```
* $$ JOB JNM=MYJOB,CLASS=Q,PRI=9,DISP=D,XDEST=OTHERVSE,TDISP=K
```

This disposition becomes effective and visible, as soon as a job or output entry is added to the transmit queue.

For Spool-Access PUT-OPEN-OUTPUT requests, the transmission disposition D|H|K|L may be provided in PWRSPL field SPLOTDP at offset X'CE'. This disposition takes effect, when the output entry is added to the XMT queue. The default transmission disposition is D. Any invalid specification is rejected by the existing PXPRETCD/FBKCD=08/0B.

Remember the following main disposition rules:

1. Disposition in a queue display of a local queue means local disposition, namely 'execution' disposition for jobs in the RDR queue or 'processing' disposition for output entries in the LST/PUN queue. The transmission disposition is invisible and cannot be altered on a local queue.
2. Disposition in a queue display of the XMT queue means transmission disposition. The local disposition is invisible and cannot be altered on the transmit queue.
3. All queue entries enter the transmit queue with their intended TDISP|SPLOTDP transmission disposition, unless overwritten by the PALTER command.
4. When jobs are read in, or output is created for the XMT queue by a specified target node name, they will enter this queue with transmission disposition H (see message 1RA1I), as soon as the node name is unknown to the originating system.
5. When queue entries are moved between transmit and local queues, the disposition of the queue just 'left' is preserved.
6. Whenever received entries enter the local queues of the final target node, the preserved local disposition becomes valid.

Drop 'Last-One' and Print Shorter Separator Page

Separator Pages are intended to be printed on continuous forms across the perforation to show the printer operator where an output ends and a new output starts. The number of separator pages is specified in the POWER macro (JSEP=n) and may be influenced by the * $$ LST statement and by operands of the 'PSTART LST,...' command. All 'n' separator pages contain the same information and are identical. The last lines of the 'n'th separator page are printed on the next page which is then filled with an additional separator page, called 'Last-One'. This 'Last-One' separator page is different and does not print across the perforation, because it contains less lines than the preceding separator pages.

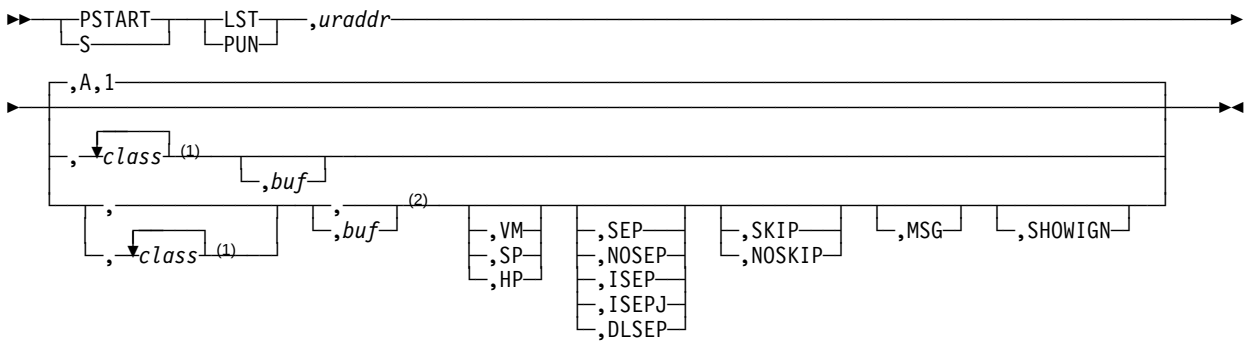
All separator pages can be forced to be equal by the ISEP SET statement of the VSE/POWER autostart or by the ISEP|ISEPJ operand of the 'PSTART LST,...' command. Then the last lines of the 'Last-One' separator page are printed on page 'n+2'.

For LASER printers, like the 3800, which can not print on the perforation or for OEM page printers, which emulate IBM impact printers but use single sheets, where no perforation exists, this VSE/POWER function is useless. The lines, which should cross the perforation, are printed at the top of the next page or sheet.

To avoid this, the operator has now the choice to start the local list task with the new operand DLSEP, which drops the 'Last-One' separator page and decreases the size of the preceding separator pages to let them fit on one page or one sheet. The size of the separator page is defined by the new VSE/POWER autostart statement 'SET **LINE=n', which is explained in 'New SET statements for VSE/POWER startup'. For a layout of a separator page refer to VSE/POWER Administration & Operation heading 'Separator Pages - Layout and Control'.

You may set DLSEP as default for each LST task using the SET statement 'SET DLSEP=YES|FORCE'.

PSTART Command for Local Writer Task



Notes:

1 You may specify up to four classes. If nothing is specified, the default class is A.

2 You must specify at least one additional operand.

Figure 70. VSE/POWER PSTART Command

DLSEP

Code this operand if you want to drop the 'Last-One' separator page for list output. Via autostart statement SET **LINE you can define the size of the separator page by specifying how many perforation lines (normally, when DLSEP is not specified, 8 identical lines, starting with '****') are printed. If 'SET **LINE=n' is not specified, a default of 4 perforation lines is printed. VSE/POWER takes the number of separator pages from the JSEP operand of the POWER generation macro. However, if JSEP=0 was specified in the POWER generation macro, VSE/POWER assumes one separator page.

When Print Output is Misaligned

Overview: The correct alignment of printed output can be achieved by specifying a Forms Control Buffer (FCB). The FCB describes the layout of the form on which the produced output will be printed by VSE/POWER lateron. The minimum information contained in an FCB is 'number of lines per page', 'position of channel 1' and for PRT1 printers also whether 6 or 8 lines per inch should be printed. The specified FCB is used during spooling time by VSE/POWER to emulate a printer with such an FCB to the output producing application. When VSE/POWER prints the output, it loads the specified FCB into the printer for correct alignment. If no FCB is specified at spooling time, the LTAB (see "Detailed description") is used instead of an FCB and the printer's default FCB is used at print time. This may cause problems, if LTAB and FCB differ.

For jobs with no FCB specified in the * $$ LST statement, correct print page layout can now be achieved by

- Including the new SET FCB=DEFAULT statement into the VSE/POWER startup procedure. This triggers usage of the spooled devices' default FCB at output spooling time and generates a match to the default FCB loaded into the real printer at output printing time.

Please keep in mind that the IBM supplied standard FCBs $$BFCB.. are designed for 12 inch forms and a line density of 6 lines per inch (lpi). If you use different forms and/or a different line density, you should regenerate the FCB. For generating an FCB refer to z/VSE System Control Statements or z/VSE Installation.

Detailed Description: VSE/POWER can serve modern FCB controlled printers as well as carriage control tape printers, whose print page layout is described by the LTAB specification of the POWER generation macro or of the * $$ LST statement.

An FCB (or LTAB) is used by VSE/POWER at two different times:

1. At **spool-time** a job is running in a partition and produces output using a printer device which is a 'spooled device' (a device controlled by VSE/POWER, defined as such when the partition is started). Then VSE/POWER intercepts the I/O started to the printer device and writes the output data to the VSE/POWER spool file. At this time the FCB is loaded and used to control the number of lines written to a page and to determine the size of a separator page.
2. At **print-time** the print output (created 'long ago') is read from the spool file and passed to a real printer started with the PSTART LST,cuu command. At this time the FCB is loaded into the printer buffer using the z/VSE LFCB macro.

Usually, the FCB used at spool time is the same one used at print time. If they are not the same, the layout of a page may not be as expected, for example, the number of lines per page will cause a splitting of pages at an unexpected position. Therefore, you should trigger the usage of the same FCB at spool and print time by explicitly specifying an FCB in a * $$ LST statement.

If you do **not** specify an FCB (nor LTAB) in a * $$ LST statement, VSE/POWER uses an LTAB at spool-time, namely the LTAB specified in the POWER macro or its VSE/POWER default. At print-time (for FCB type printers) VSE/POWER uses the system default FCB which is printer-type dependant. If the LTAB does not reflect the default FCB's specifications, it may lead to an incorrect layout of the printed pages.

To avoid incorrect layout, tell VSE/POWER not to refer to the LTAB but to use the printer's system default FCB already at spool time by placing the

```
SET FCB=DEFAULT
```

statement into the startup procedure.

FCB=DEFAULT|DEFAULT,PRT1=xx

The operand causes VSE/POWER to use a default FCB when controlling the spooling of list output instead of the LTAB specified in the POWER macro. This affects only list output spooled by a job running in a partition if neither the FCB nor the LTAB operand has been specified in a * $$ LST statement.

This operand does not apply to list output spooled via the Spool-Access support. This output is written to the VSE/POWER spool file by **not** using any FCB. If an FCB has been specified within the SPL, this FCB is used only at print-time.

The default FCB is selected as described in chapter "System Buffer Load" in "z/VSE System Control Statements".

Printer Type	FCB Name
1403	n.a.
3211 (PRT1)	$$BFCB
3203-5 (PRT1)	$$BFCB00
3289-4 (PRT1)	$$BFCB10
3262 (PRT1)	$$BFCB22
4245 (PRT1)	$$BFCB23
4248 (PRT1)	$$BFCB
4248 (Native) and 6262	$$BFCBWM

For example, the default FCB for a 6262 printer or a native 4248 printer is $$BFCBWM. As printers of different type (for example 3211, 4245, 3203, 3289, 3262) are added during IPL time with the same device type PRT1, the operand PTR1=xx may be specified to define the device specific default FCB. The xx are the two last characters of the device specific default FCB, for example the default FCB for a 4245 printer is $$BFCB23 and therefore PRT1=23 has to be specified.

Notes:

1. Only one default FCB for PRT1 printers may be specified. If printers of two different device types have been added as PRT1, the same default FCB is used no matter which printer is currently used for the list output.
2. If the PTR1=xx operand is omitted, $$BFCB is used for printers added as PRT1.
3. The PTR1=xx operand does not influence the default FCB for printers added with a different device type code than PRT1.
4. If the default FCB is loaded, but VSE/POWER detects some error with this FCB, the message 1Q54I is issued and the LTAB is used as specified in the POWER macro.

VSE/POWER Spooling Considerations

Important VSE/POWER Specifications for Output Spooling

In the following, the relationship between certain specifications for VSE/POWER Output Spooling are explained. For simplicity reasons, this is done only for LIST output; but the same rules, considerations and specifications are also valid for PUNCH output.

A) Specifications to be done for OUTPUT Spooling

During execution of a job in a 'VSE/POWER controlled partition', output is produced using the following rules:

1. In your program you specify the logical unit to which the output will be written. For example, by:

   ```
   DTFPR DEVADDR=SYSLST
   ```

2. With the // ASSGN statement of your job, you direct the output from the logical unit to a physical unit of your choice. For example:

   ```
   // ASSGN SYSLST,F9E
   ```

3. If the output should be spooled by VSE/POWER, this physical unit must be specified in the PRINTERS=cuu,cuu.... autostart statement for the corresponding partition. For example:

   ```
   PSTART BG,...
   READER=...
   PRINTERS=...,...,F9E,...
   PUNCHES=...
   ```

4. With the * $$ LST statement, you can specify several output attributes for your output entry (like JNM, DISP, PRI, etc.). With the LST=cuu operand of the * $$ LST statement, you tell VSE/POWER that this * $$ LST statement should be taken for the output spooled to the cuu you specified in the LST=cuu operand. For example:

   ```
   * $$ LST JNM=...,...,LST=F9E,...
   ```

   a) * $$ LST statement without LST=cuu operand

   If the LST=cuu operand is omitted in the * $$ LST statement, the attributes specified in this statement are taken for the output spooled to that printer device which is specified **first** in the sequence of PRINTERS=**cuu**,cuu,... .

b) no * $$ LST statement at all

if you are spooling output to a printer device, for which no * $$ LST statement is provided, then this output entry gets default output attributes (like CLASS=A,DISP=D,PRI=3,etc).

Note: You may request VSE/POWER to warn you by console message

---

1Q8CI DEFAULT OUTPUT VALUES USED FOR *jobname jobnumber*, **SPOOLED DEVICE** *cuu*

whenever default output attributes are assigned to output being spooled - just place the NTFY=(*,R000) operand into the * $$ JOB statement of the questionable job to request notification of the central operator!

B) How to Link the OUTPUT Specifications

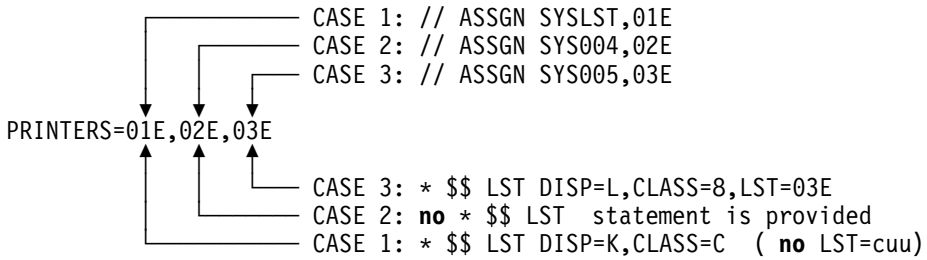
Respect the following rule:

Your output obtains only the desired * $$ LST attributes, if the spooled printer device ...

- selected by the ASSGN statement and
- addressed by your * $$ LST statement

... is the same!

In the following there are three cases explained:



Case 1 :

- Your program writes output to SYSLST
- SYSLST is assigned to 01E
- 01E is specified as **first** printer in PRINTERS=01E,02E,03E chain.
- The LST=cuu operand is omitted in the * $$ LST statement, therefore this statement applies to the **first** of PRINTERS=01E,02E,03E , namely to 01E.
- **Result:** The output entry spooled to 01E gets the specified output attributes DISP=K,CLASS=C.

Case 2 :

- Your program writes output to SYS004
- SYS004 is assigned to 02E
- 02E is specified in PRINTERS=01E,02E,03E
- **no** * $$ LST statement is provided at all
- **Result:** The output entry spooled to 02E gets the default output attributes DISP=D,PRI=3,CLASS=A,etc.

Case 3 :

- Your program writes output to SYS005
- SYS005 is assigned to 03E
- 03E is specified in the PRINTERS=01E,02E,03E .
- By the LST=03E operand, the * $$ LST statement applies to the output entry spooled to printer 03E.
- **Result:** The output entry spooled to 03E gets the specified output attributes DISP=L,CLASS=8.

C) Specification Errors that Lead to the 'Ignored * $$ LST Statement' Sympton

For the following cases we assume that your program produces output for SYS004:

1. **Case 1:** The LST=cuu operand is omitted on the * $$ LST statement.

   Specifications:

   ```
   - PRINTERS=FEE,F9E,F8E
   - * $$ LST JNM=...,DISP=K,PRI=1
   - // ASSGN SYS004,F9E
   ```

   Result: The output entry spooled to F9E obtains the default output attributes. The provided * $$ LST statement is valid for device FEE because of the missing LST=F9E specification.

2. **Case 2:**The output is directed to another print device than specified in the LST=cuu operand.

   Specifications:

   ```
   - PRINTERS=FEE,F9E,F8E
   - * $$ LST JNM=...,DISP=K,CLASS=X,LST=F8E
   - // ASSGN SYS004,F9E
   ```

   Result: The output entry spooled to SYS004 obtains the default output attributes because the * $$ LST statement is valid for F8E (LST=F8E), but output is spooled to F9E.

TCP/IP Connection to RSCS

RSCS Specifications for NJE Communication to VSE/POWER via TCP/IP:
VSE/POWER 6.5 (running on VSE/ESA 2.5) and all later releases exchanges data with remote systems via the NJE (Network Job Entry) TCP/IP protocol in addition to the BSC, SNA or virtual CTC protocol. The remote host can even be a VM/ESA system running RSCS V3.2 or a later release.

Once the TCP/IP connection to an RSCS node has been successfully established, the connection may stop every two hours, and restarts thereafter automatically. This happens because VM/ESA sends some keep-alive packets, which can not be understood by all hosts, as it is described in *VM/ESA TCP/IP Function Level 320 Planning and Customization*, SC24-5847, (see paragraph *Usage Notes* of the KEEPALIVEOPTIONS statement). These keep-alive packets are sent every nn minutes, where nn is the value specified for the INTERVAL parameter of the KEEPALIVEOPTIONS statement. If no value is specified, the connection stops every 2 hours (default is 120 minutes).

In order to avoid that the TCP/IP connection to an RSCS node stops, the sending of the non-understandable keep-alive packets must be suppressed. This can be achieved in one of the following ways:

1. Specify "SENDGARBAGE TRUE" in the KEEPALIVEOPTIONS statement used by VM/ESA TCP/IP (see KEEPALIVEOPTIONS statement in *VM/ESA TCP/IP Function Level 320 Planning and Customization*, SC24-5847).
2. Specify "KEEPALIVE=NO" in the PARM statement used by RSCS to define a TCPNJE-type link (see PARM statement in *VM/RSCS Networking Planning and Installation*, SH24-5219).

Chapter 7. Librarian

You can use the z/VSE Librarian via these interfaces:

- Command Interface (EXEC LIBR). This is the interface used by most users.
- Macro Interface (LIBRM, LIBRDCB). Used by programs to access libraries.
- Internal Macros (used by Linkage Editor, IUI, DUMP, INFOANA and all LIBR functions). Used mostly by IBM programs.

Librarian Return Code Conventions

- Commands
  - Return code 0 : function completed
  - Return code 2 : indicates result (COMPARE: notequal; TEST: library damaged).
  - Return code 4 : function = noop (e.g. delete non-existent member)
  - Return code 8 : function (partially) failed
  - Return code 16: function abended

  With every return code > 4 there is an additional message which says what exactly has failed. These messages are normally on SYSLST.
- Macros (LIBRM)
  - R15 = 0 : function completed
  - R15 = 4-12 : depends on function (see System Macro Reference)
  - R15 = 16 : external error; Feedback code is in R0.
  - R15 = 20 : internal error; Feedback code is in R0.
  - R15 = 32 : security violation.

  With every r15 > 12 there is an additional message which says what exactly has failed. IALCMSG points to this message.
- internal Macros
  - R15 = 0 : function completed
  - R15 = 4-12 : depends on function
  - R15 = 16 : external error; Feedback code in INLCLRPL or LBRACCDS.
  - R15 = 20 : internal error; Feedback code in INLCLRPL or LBRACCDS.
  - R15 = 32 : security violation.

  With every R15 > 12 there is an additional message which says what exactly has failed. This message is returned in areas provided by the user in the INLCLAMB.

IUI suppresses Librarian messages.

Librarian commands normally ABEND after return code 16 or 20 with a DUMP (DUMP is always provided after Message L151/L152).

How to Analyze What Went Wrong

- Check which interface is used:
  - EXEC LIBR
  - Linkage Editor
  - IUI
  - MSHP
  - Info/Analysis
  - JOB CONTROL (e.g. LIBDEF/LIBDROP)
  - vendor programs
  - own applications which use LIBRM.
- Carefully check return codes and messages (normally on SYSLST).
- If a feedback code was given check the Librarian Diagnose Reference Manual (DRM).

Important Messages

- L151I UNEXPECTED RETURN CODE FROM MACRO/MODULE nnn IN MODULE mmm PHASE xxx - RC = nn

  Check if MACRO/MODULE mentioned is a supervisor service (e.g. System Macro Reference). Often this message is caused by damaged libraries or overwritten control blocks. This message causes a dump.
- L152I ENTRY CONDITION FOR MODULE nnn IN PHASE xxx FAILED - FEEDBACK CODE = nn

  The Feedback code specifies the reason of the error is described in the Librarian DRM or (partially) in SYSTEM MACRO REFERENCE. Often this message is caused by damaged libraries or overwritten control blocks. This message causes a dump.

Normal action is to run a TEST L=lib on the accessed library. If the library is damaged, the dump will not provide any additional information. Repair or recreate the library as soon as possible.

Recreation of Libraries

If one of the following conditions occurs, you may suspect that a library is damaged or in an inconsistent state:

- Messages L157 or L158 are displayed.
- FETCHing/LOADing a phase fails either with an error message or return code indicating retrieval error.
- Messages L150, L151, L152 are displayed. Although these messages are not directly triggered by library damages, you should verify that the libraries involved are in a consistent state.
- You get a 'library full' message even if the command LISTDIR LIB displays FREE SPACE.
- The number of sublibraries displayed with LISTDIR LIB does not match the number of sublibrary names listed.

- The number of members displayed with LISTDIR SUBLIB is not identical to the number of member names listed.

You may use the TEST command to verify the correctness of a library, sublibrary, or of an individual member.

What has to be done to repair a library ? If a part of a library is damaged always recreate the whole library, especially if any library block chains are damaged.

A library can be recreated as follows:

Library in BAM Space

```
BACKUP L=lib
DEFINE L=lib REPLACE=YES
RESTORE L=lib
```

Figure 71. Recreate library in BAM space

Note: BACKUP tries to back up all valid members. If it fails, an previously created backup-tape must be used for RESTORE.

Library in VSAM Space

```
BACKUP L=lib
DELETE L=lib
DELETE CLUSTER
DEFINE CLUSTER
DEFINE L=lib REPLACE=YES
RESTORE L=lib
```

Figure 72. Recreate library in VSAM space

Note: The step DELETE/DEFINE CLUSTER is recommended but not required.

DEFINE L=lib REPLACE=YES: Creates a new library without checking the old contents.

Problems might occur with VSAM managed space, if DELETE CLUSTER has been done before DELETE LIBRARY. After DEFINE CLUSTER the DEFINE LIBRARY might bring unpredictable results.

If a problem occurs with this sequence the library descriptor must be destroyed by DITTO. The library descriptor is located at the beginning of the first extent of the library and can be identified by the keyword 'LIBRARY' followed by the library name. Its length is 1024 bytes.

BACKUP Abends or Loops

In general BACKUP tries to back up all valid members. BACKUP might abend if the source library is damaged. The cause is normally a damaged library block chain in member data.

Example: Damaged member A.PROC is in OLDLIB.S1. As DELETE might not work, RENAME the defective member e.g. into ZZZ.ZZZ, so it will be the last member in the alphabetical order. Then copy all other members step by step into a

new (temporary) library.sublibrary and copy all other sublibraries contained in the damaged library into the temporary library. Afterwards recreate the damaged library and copy all sublibraries back.

```
ACCESS S=OLDLIB.S1
RENAME A.PROC ZZZ.ZZZ                  /* rename defective member  */
CONNECT S=OLDLIB.S1 : NEWLIB.HELP      /* NEWLIB.HELP must exist   */
COPY A*.*,B*.*  ...             /* copy all members except ZZZ.ZZZ */
              /* COPY or RESTORE other sublibs of OLDLIB to NEWLIB */
DEFINE L=OLDLIB REPLACE=YES
                                /* RESTORE/COPY sublibraries back  */
```

Figure 73. BACKUP library with defective member

BACKUP LIB/SUBLIB tries to skip members or sublibraries in error. It informs you through the messages L080 and L081 which sublibraries or members are skipped due to retrieval failures.

Limitation: BACKUP is able to skip defective members consisting of up to 64 library blocks.

Defective members larger than 64 library blocks cannot be skipped by BACKUP, even if the partition size is increased. In this case you have to delete the member in error before restarting the BACKUP run. This restriction does not apply if the error is detected within the first 14 library blocks. In this case, BACKUP has not written any member data onto tape when it detects the error and, therefore, is able to skip the defective member.

RESTORE of Libraries Not Uniquely Assigned or Shared by Processors

You cannot restore libraries for which LIBDEF statements or Librarian commands (e.g. ACCESS) are active or are in use by two or more processors.

You have to deactivate the libraries by issuing LIBDROP statements and by terminating Librarian commands in all processors accessing these libraries. Failing to do so causes abnormal termination of all tasks retrieving members from these libraries after RESTORE since the in-storage information is kept for the LIBDEF statements or Librarian commands is not updated automatically in the other processors. The RELEASE SPACE command does not help in this situation since it does not update all in-storage information which might have been updated by RESTORE LIB.

You can, however, restore sublibraries which are attached to the system either through LIBDEF statements or Librarian commands. (Of course, you have to specify REPLACE=YES on the RESTORE command.)

The REUSE value of the sublibrary on disk will be maintained, that is, it will not be overridden by the REUSE value of the sublibrary on tape.

Replacing an existing sublibrary is done in the following system-internal steps:

1. All members in the existing sublibrary are deleted. The library blocks of the index are put unchanged into the space reclamation chain.

   For sublibraries with attribute REUSE=IMMEDIATE, the member space is freed immediately and available for the sublibrary to be restored.

For sublibraries with attribute REUSE=AUTOMATIC which are shared by partitions or processors, the member space is put into the space reclamation chain of the sublibrary and therefore not available for the sublibrary to be restored. In this case, the free space of the library must be large enough to receive the contents of the new sublibrary.

For sublibraries with attribute REUSE=AUTOMATIC which are not shared by partitions or processors, the member space is freed immediately and available for the sublibrary to be restored.

2. The date in the sublibrary directory entry is updated. (The entry in the sublibrary directory is not removed.)
3. The new members are cataloged into the sublibrary and a new member index is built.
4. A fresh second-level directory (SLD) is built in the processor performing the RESTORE action if the sublibrary shows up on LIBDEF PHASE statements. The library control statements in the SVA are updated (specifically, the pointer in the sublibrary definition table entry pointing to the library block of the highest level of the member index).

As you see, there is a time slice when neither the old nor the new members may be found. Therefore, you have to ensure (by administrative actions) that programs relying on the availability of either the old or the new version of the members do not run concurrently to the RESTORE SUBLIB command.

For sublibraries with attribute REUSE=IMMEDIATE, you have to make sure that no local directory lists point to phases in the sublibrary being restored.

For sublibraries with attribute REUSE=AUTOMATIC, local directory lists continue to point to the old versions of the phases.

For sublibraries showing up on LIBDEF PHASE statements, SLDs are maintained in-storage. SLDs are updated automatically in the processor restoring the sublibrary.

Shared libraries: If a sublibrary restored in one processor is shared by other processors, you must perform, for libraries showing up on LIBDEF statements, the RELEASE SPACE command in all processors accessing the sublibrary.

RESTORE OLDLIB

Attention: Old tapes are often damaged and no longer readable.

Libraries in old format (pre VSE/SP V2) may contain members starting with /. Todays Librarian cannot process these members.

Restore IJSYSRS - IJSYSRS corrupted

Restore from a stand-alone backup

If you have a stand-alone backup available on a physical tape volume, that is, you have made a backup as follows:

```
LIBR BACKUP LIB=IJSYSRS,RESTORE=STANDALONE,TAPE=cuu
```

In this case you can IPL the tape (IPL cuu) and restore the IJSYSRS library.

Restore from a virtual tape

If you have the backup on a virtual tape, you need a second running z/VSE system for the restore.
In this case you can ADD your corrupted DOSRES (dosres_failcuu) to a running z/VSE system and then do the restore from the virtual tape.
To do so:
IPL your system with: *ipl_device LOADP ..P.*
When you are prompted enter: *STOP=ADD*
When prompted enter: *ADD dosres_failcuu,type*

Note: The dosres_failcuu must be higher than the cuu of your IPLed DOSRES device.
Sample job to restore IJSYSRS from virtual tape:

```
* $$ JOB JNM=SYSREST,DISP=D,CLASS=0
// JOB RESTORE IJSYSRS FROM BACKUP VIRTUAL TAPE
// DLBL IJSYSR1,'VSE.SYSRES.LIBRARY',99/366,SD
// EXTENT SYS009,DOSRES,1,0,1,899
// ASSGN SYS009,dosres_failcuu
// VTAPE START,UNIT=480,LOC=...,FILE='....',READ
// PAUSE - RESTORE LIBRARIES
// MTC REW,480
// EXEC LIBR,PARM='MSHP'
   RESTORE LIB = IJSYSRS:IJSYSR1,REPLACE=YES,DATE=OLD,TAPE=480
/*
VTAPE STOP,UNIT=480
/&
* $$ EOJ
```

What To Do In Error Cases?

Unexplainable behavior, hardwait, program check or loop: apply TEST command to accessed/updated library after the system has been recreated. If a library is damaged (as indicated by the result of the TEST command) a BACKUP should be tried. If unsuccessful an older backup version has to be used for RESTORE.

Test of Libraries

To test the condition of a library issue this librarian command:

```
TEST L=lib
```

A test may run up to two hours for a large library.

Error conditions:

- Error Codes (described in DRM)

402 uncritical, can be repaired by TEST L=xxx REPAIR=YES
401 critical, BACKUP still possible, if this is the only fault.
056 logical consequence of 402 or 401.
other restore library from available backup.

- Test command loops: library block chains are damaged, restore library from an available backup
- Test command abends or program checks: restore library from an available backup

A test on a sublibrary (TEST S=lib.sublib) does only a subset of the tests that a test on a library does, so do not use it for problem determination.

If the TEST run produces the error '056' or '402', you may rerun the TEST LIB command with REPAIR=YES to resolve these error situations. Note that REPAIR=YES requires that

- the library is not used by any other task,
- the library is not specified on LIBDEF statements,
- the library does not reside on a DASD volume which is shared by processors (defined by the SHR parameter on the IPL ADD command).

What Can Damage a Library?

A library can be damaged by any forced interruption of the Librarian during write access (CANCEL with FORCE, hard wait, program check), or by any simultaneous update in a shared environment.

Error 402 might occur after an ABEND of any function writing or updating a member. Normally the Librarian (Commands and LIBRM) cleans up the library before abnormal termination - DUMP/INFOANA cleans up since VSE/ESA 1.2/1.3.

Other Problems

Member Not Found

You might detect one of these symptoms:

- LD *.* shows a member but LD membername.membertype not.
- A FETCH doesn't find an existing phase.

Reason: Two or more z/VSE's share a disk (shared library). The library was updated on System A. System B had some information about this library in storage (LIBDEFs, SDL entries etc.). These are not updated yet.

If a sublibrary is shared by processors and members of it cannot be found by one processor (or the old versions are still found). It is likely that the in-storage information of this processor is obsolete due to member index modification in other processors.

Solution: Run RELEASE SPACE for the updated library on every machine. Attention: during RELEASE SPACE no task should access this library, it might run into unpredictable results

The command RELEASE SPACE updates the system in-storage information (but not local directory lists).

Influences on Number of SIOs

A sublibrary is considered as large if it contains a large number of members, the member index of a large sublibrary consists of more index levels, each one causing a SIO when locating a member name in the sublibrary.

Note: Due to the member index structure (B-tree), all members in a specific sublibrary are located with the same number of SIOs or with a number which differs by one.

Using small sublibraries reduces the number of SIOs for locating a member in a sublibrary. However, you may have to increase the LIBDEF SEARCH chain, thus increasing the number of SIOs for going through the SEARCH chain.

Since the sublibraries of the LIBDEF PHASE SEARCH chain get in-storage second-level directories (SLDs), the size of a sublibrary generally does not impact the performance of the supervisor FETCH/LOAD-phase function or the overall system performance. However, you may notice a performance improvement when working with small sublibraries for all Librarian actions which modify the member index, for example COPY, RESTORE, RENAME or CATALOG.

With these contradictory aspects in mind, it is generally advisable to group members into sublibraries by application, product versions, vendors, etc. without caring much about sizes. An exception is if the above-mentioned Librarian commands are considered performance critical. Then you should choose small sublibraries.

The following figures give an idea of what is considered a small and large sublibrary in this discussion:

- small: up to about 300-420 members

- large: more than about 9000-125000 members

Sublibraries between the above ranges are considered medium-sized and will require 2-3 SIOs to locate a member name. Only rough figures can be provided here since the member index size depends on the filling up of the index library blocks, the number of different member types in a sublibrary, or the percentage of phases in the sublibrary (phases have larger entries in the member directory the other member types). The lower values apply to sublibraries containing only phases, the upper values to sublibraries containing non-PHASE member of predominantly the same type.

How to Find Out the Number of Member Index Levels of a Sublibrary

Use the Librarian command TEST SUBLIB=lib.sublib. A line near the end of the display output tells you the NUMBER OF INDEX LEVELS.

The number of member index levels reflects the number of SIOs needed to locate a member in a sublibrary. The number of SIOs is either identical to the number of index levels or one higher. However, the number of index levels does not affect the performance of the supervisor FETCH/LOAD-phase function since in-storage second-level directories (SLDs) which allow locating a phase in a sublibrary with just one SIO are provided.

This value is not displayed on the LISTDIR output since it is not considered important for most installations and might cause confusion for programmers unfamiliar with Librarian internals.

One Large Extent versus Several Smaller Extents

The Librarian allows you to define a private library with up to 16 extents in BAM controlled space and up to 32 extents in VSAM managed space.

For BAM controlled space, you have to specify the individual extents directly in the DLBL/EXTENT statements. For VSAM managed space, you have influence on the extent sizes through the primary and secondary allocation values when defining the VSAM cluster. The following evaluation should shed some light on the factors affected by the number of library extents:

- Opening a library: Each library extent is opened individually, that means, for each extent there is an invocation on the OPEN routines (this applies to libraries in BAM controlled space as well as in VSAM managed space). The effect is that the elapsed time for opening a library increases with the number of its extents. However, libraries which are already known to the system will not be re-opened again, for example, a library specified on a LIBDEF statement will not be re-opened in a processor as long as this LIBDEF statement is active (not LIBDROP'ed). Thus, the time for opening a library is not considered performance-critical for most installations. The picture changes for libraries which do not show up on a LIBDEF statement and are frequently accessed online through LIBR commands. In this case, the overhead for opening additional library extents may be noticed when, for example, a Librarian ACCESS command is entered.
- Start I/O's (SIOs): Once a library is open, the number of SIOs to the data (index as well as member data) is independent of the number of library extents. Spreading library extents on different volumes may give you the opportunity to help balance the channel/device activities. Whether you gain a noticeable

performance increase by doing so depends on how often the library data on these extents is accessed and on the other data traffic on the channel and device. Having only one large library extent on a volume instead of several smaller ones may or may not decrease the average SEEK time for data on this volume depending on the pattern of accesses to the library data and the other data on this volume. The Librarian does not offer a direct way to put selected sublibraries or members into a specific library extent. Usually, it will not be worth the effort of trying to balance the channel/device workload based on the accesses to library data. The accesses to non-library data sets probably have a greater influence.

- Path length: The only time the Librarian routines (and supervisor FETCH/LOAD) are confronted with multiple extents is when building the CCW chain for accessing library data. The path length needed by the Librarian routines (and supervisor FETCH/LOAD) to translate a logical address into its physical DASD address increases only slightly with the number of library extents. You will hardly notice the difference.

So the conclusion of the above discussion is: A higher number of library extents causes a noticeable performance decrease only in rare situations.

Space Reclamation Attributes AUTOMATIC versus IMMEDIATE

With the parameter REUSE= in the DEFINE SUBLIB command you select one of the two space reclamation algorithms provided by the Librarian. The algorithms are bound to individual sublibraries, not to entire libraries.

With attribute AUTOMATIC (default) local directory lists remain valid even if the corresponding phases are deleted/replaced. Also, long-running procedures will come to a normal end even if the are deleted/replaced concurrently to their execution. This system behavior is achieved by postponing the reuse of deleted/replaced member space of the sublibrary is not in unique use by one task/partition/processor when members are deleted/replaced.

Attribute IMMEDIATE makes the space of deleted/replaced members available for immediate reuse, even if the sublibrary is shared by several tasks or partitions or processors. Thus, programs with local directory lists or long-running procedures may terminate abnormally if the corresponding phases or the procedures in execution are deleted/replaced concurrently to their execution. However, integrity is not exposed since the system is able to detect library blocks which are reused for another member.

AUTOMATIC and IMMEDIATE are entirely identical in their effects:

- if the sublibrary is by only one partition/task at the time members are deleted/updated/replaced

  and

- if the library (containing the sublibrary) does not reside on a volume which is shared by two or more processors.

If you want to change the attribute for an existing sublibrary, you may do so anytime by using the CHANGE SUBLIB command. When changing from AUTOMATIC to IMMEDIATE, data library blocks in the space reclamation chain are freed.

Based on the above discussion, you may prefer the attribute

- AUTOMATIC for sublibraries used for production runs,
- IMMEDIATE for sublibraries used for program development.

What Does the RELEASE SPACE Command Do ?

The RELEASE SPACE command performs the following actions for individual sublibraries (specification SUBLIB=lib.sublib) or for each sublibrary of a library (specification LIB=lib) in the following order:

1. It invalidates the phases in the space reclamation chain, thus preventing supervisor FETCH/LOAD from loading them into partitions. That means, in-storage directory entries can no longer be used to retrieve such phases.
2. It adds the library blocks of the space reclamation chain to the free space inventory, thus making these library blocks available for immediate reuse (the DELAYED SPACE/DELAYED LB'S values displayed by the LISTDIR command will be set to zero increasing the FREE SPACE values). The contents of the released library blocks remain unchanged except for phases (phases are invalidated).
3. It updates the second-level directory (SLD). (SLDs are built only for sublibraries showing up on LIBDEF PHASE or LIBDEF * statements.)
4. It updates the in-storage pointer (saved in the sublibrary definition table in the SVA) to the library block containing the highest level of the member index (this pointer is used to locate members in the sublibrary for retrieval).

If a sublibrary is shared by two or more processors, it is strongly recommended that you execute the RELEASE command in all processors accessing the sublibrary and not just in one. You are asked to do so by the messages L047/L048 which are displayed on SYSLOG. The reason for this is that the above actions have to be performed in all processors, even if the space reclamation chain is empty after execution of the first RELEASE command.

For sublibraries showing up on LIBDEF PHASE statements you must refresh in-storage information in other processors:

1. Terminate programs with local directory lists pointing to replaced phases.

   Execute the Job Control command SET SDL if phases in the SVA have to be replaced or if SDL pointers point to replaced phases in the system sublibrary IJSYSRS.SYSLIB.

   Ensure that this action is performed in all processors accessing replaced phases before continuing.
2. Run the Librarian command RELEASE SPACE in all processors accessing the sublibrary to update in-storage information (SLD, pointer to highest member index library block).
3. Execute the Job Control command SET SDL if phases in the SVA have to be replaced or if SDL pointers point to replaced phases in the system sublibrary IJSYSRS.SYSLIB.
4. Restart programs with local directory lists.

If you fail to adhere to the above sequence of steps, you risk that tasks may terminate abnormally due to 'member not found' or 'member space reused'

conditions. Executing the RELEASE command (even in an incorrect step sequence) cannot destroy a library/sublibrary or cause integrity exposures.

Shared Libraries

Update of In-Storage Information for Shared Libraries

The following information about sublibraries is kept in virtual storage and subject to sublibrary changes:

```
- SLD (second-level directory)                    owned by system
- Pointer to the highest level of member index    owned by system
- System Directory List (SDL) pointing to phases
     in the system sublibrary IJSYSRS.SYSLIB      owned by system
- Local Directory Lists                           owned by program
```

SLDs are built only for sublibraries which show up on LIBDEF PHASE statements (which include LIBDEF * statements). They are used only by supervisor FETCH/LOAD to retrieve phases.

The pointer to the highest level of the member index is used to locate e.g. object modules for link-editing, source books for compilations, and procedures for EXEC PROC= execution. The Librarian uses this pointer for retrieval of members of any type (e.g. LIST, LISTDIR, PUNCH, BACKUP, COMPARE, COPY).

Updating local directories is always the user's responsibility (e.g. CICS-PPT via NEWCOPY). The SLD, the pointer to the highest member index level, and the SDL pointer are updated automatically by the system in the processor performing the sublibrary change.

If a sublibrary is shared by processors and its contents are changed, its in-storage information might have an old status in all processors except the one performing the change. The system-owned information is updated by the following actions in the processor performing the action:

- The SLD is updated automatically whenever supervisor FETCH/LOAD uses more than one SIO to locate a phase in a sublibrary of a permanent LIBDEF PHASE chain.
- The Librarian command RELEASE SPACE updates the SLD and the pointer to the highest member index level.
- If you detach the sublibrary from all partitions (LIBDROP) and attach it again (LIBDEF), a new SLD is built and the pointer to the highest member index level is refreshed.
- If you catalog, delete, rename members (of any type), the SLD and the pointer to the highest member index level are updated.
- The Job Control statement SET SDL updates the SDL pointers and phases in the SVA (using the LOAD macro).

Is There a Performance Impact if Libraries Are Shared by Processors?

For retrieval of members there is no performance impact if libraries are shared by two or more processors. The actions performed by Librarian routines and supervisor FETCH/LOAD are exactly the same as for non-processor-shared libraries (member retrieval does not require locking).

For library functions which store members in processor-shared libraries you may notice a performance decrease compared to storing in non-processor-shared libraries.

The reasons are two-fold:

- Locking of resources (for example, the free space inventory) has to be done across processors using the lock file. This increases the path length for the locking service and causes read/write SIOs to the lock file.
- When storing members in libraries which are neither shared by processors nor by other tasks/partitions, the Librarian keeps the free space inventory in partition storage, thus saving the read/write SIOs for updating it. This optimization cannot be done for processor-shared libraries.

Specifically the following Librarian functions may run faster for libraries which are neither shared by processors nor by other tasks/partitions:

```
RESTORE LIB/SUBLIB
COPY    LIB
MOVE    LIB
```

Since member-retrieval actions usually far outweigh member-cataloging actions, the overall system performance is unlikely to be affected by processor-shared libraries.

Note: Libraries are considered to be shared by processors when the first (or only) extent of a library resides on a volume which is shared (defined by the SHR parameter on the IPL ADD command). The Librarian treats such libraries as processor-shared regardless of whether they are used by other processors or not. Therefore, you may put the first extent of a multi-extent library on a non-shared volume to get better performance for the above Librarian actions in case a library is not actually shared by processors, but for DASD space reasons has to reside on a processor-shared volume.

Can I Share Libraries Without Having a Lock File?

If you share libraries between real processors or z/VSE guest machines running under VM, it is highly recommended that you

- specify the SHR parameter on the z/VSE IPL ADD command when defining the DASDs with the shared libraries,
- define or reference the lock file via the z/VSE IPL DLF command,
- attach the DASDs with the shared libraries for z/VSE guest machine running under VM in read/write access mode.

Without a lock file, the write SIOs to shared libraries from the sharing processors or z/VSE guest machines will not be serialized, thus destroying your libraries.

Thus layout is recommended even in situations when a real processor or a z/VSE guest machine accesses such libraries exclusively for member retrieval and never

runs library update jobs. The technical reasons for this recommendation are the following:

- Automatic space reclamation: Even if no library update jobs are submitted in one of the sharing processors or z/VSE guest machines, this processor or z/VSE guest machine may perform automatic space reclamation for a shared library causing write SIOs to this library. Without a lock file, you risk destroying these libraries. When the shared DASD is in 'read-only' status under VM, the write SIOs for space reclamation will be rejected causing abnormal termination of the z/VSE task triggering space reclamation. This can happen also at IPL time when LIBDEF statements are processed.
- Abnormal task termination: When a library is shared but its DASD is not added with parameter SHR, member retrieval tasks as well as library update tasks may be terminated abnormally with 'defect in library detected' or 'member not found' messages although the library is in good shape. A possible cause for these messages is dynamic extension of a library in VSAM managed space performed by another processor or z/VSE guest machine. For non-shared libraries, the system assumes that such a library is defective rather than verifying if the library was extended by another processor or z/VSE guest machine. The task is terminated abnormally when attempting to access the extended part of the library.
- No warning messages: You will not see any warning messages before a library or sublibrary is deleted if the system is not aware of the shared status.

In brief: You avoid a lot of potential problems by defining libraries as shared when they are actually shared.

Are Libraries Extended Automatically if They Are Shared?

Libraries in VSAM managed space are extended automatically even if they are shared and concurrently accessed by several processors. Libraries in BAM controlled space are never extended automatically.

Any of the shared processors can trigger the extension of a shared library and work immediately with the additional extents obtained. The other processors are not informed about the extension directly. They continue to work with the original library extents saved in their SVA until one of the following happens:

1. A Librarian service finds a disk address in the extended library which points outside the library extents known to the processor. This triggers an update of the library extent information in the SVA of the respective processor. This update may be initiated by a read-from-library or a write-into-library request. Thus, the added extents are automatically known to the other processors.
2. Supervisor FETCH/LOAD detects a disk address which points outside the range of library extents known to the processor. It will either cancel the task with message 0P92I or return to the invoking routine with return code 12 depending on the value of the RET parameter on the FETCH/LOAD macro. To recover from this situation, the operator should run the Librarian command TEST LIB=lib AREA=SPACE for the library indicated by message 0P92I or displayed by the failing routine. The TEST command will update the library extent information in the SVA when getting a disk address outside the known range of library extents.

This last described situation can happen only when libraries of LIBDEF PHASE chains have been extended in other processors and the added extents have not yet been accessed by Librarian services in the other processors.

To prevent such a situation from happening, the system displays the message L274 on the operator console whenever a shared library is extended automatically. This message informs the operator about the library extension and asks him to run the Librarian command TEST LIB=lib AREA=SPACE in all other processors which share the extended library (the TEST command with AREA=SPACE is recommended because it takes only a short period of time to execute it).

The library extent information in the SVA is refreshed automatically only for libraries which reside on DASD volumes added with parameter SHR at IPL time. For other libraries, it is assumed that a library is defective when disk addresses are pointing out of the range of the library extents recorded in the SVA. In this case the task will be terminated abnormally with a Librarian message.

Library Full

Check via

```
LD L=lib OUTPUT=STATUS     or
TEST L=lib AREA=SPACE
```

if there is any delayed space in the library. If so run RELEASE SPACE to free this.

Run TEST L=lib to check for 402 errors. Run

```
TEST L=lib REPAIR=YES
```

to remove errors 402. The library must be uniquely assigned.

What To Do if Library Space Is Exhausted ?

When you get one of the messages L201, L268, or L278, you should check the library status before taking steps to expand the library space:

1. Execute the Librarian command LISTDIR LIB=lib OUTPUT=STATUS and check the number of free library blocks in the list output line 'FREE SPACE'. This value must be large enough to take in the new member including a couple of library blocks possible needed for index update (when replacing a member, the Librarian first catalogues the new version before freeing the space of the old version).

   When replacing members, you may decide to delete the existing members before cataloging the new ones.

2. If you can reasonably assume that the new member should fit into the available free space based on the LISTDIR output values, then the library may be in an inconsistent state. Execute the librarian command TEST LIB=lib REPAIR=YES to check the consistency of the library and to free library blocks marked as occupied in the free space map in spite of not belonging to a cataloged member or to an index. This situation may show up as the result of a prior system break-down or CANCEL FORCE command.

   Be aware that the REPAIR function of the TEST command can only be executed if the library is not specified on LIBDEF statements and is not shared by processors. The TEST command with REPAIR=NO can be executed any time, but should not be executed concurrently to a library update. Otherwise,

TEST may display error messages based on temporary inconsistencies due to the update process.

3. Check the output line DELAYED SPACE of the LISTDIR command. If the DELAYED SPACE value is not zero, execute the Librarian command RELEASE SPACE to free this space and to make it available for reuse.

   Be aware the the RELEASE command has to be executed in all processors accessing the library and that local directory lists should be refreshed if still pointing to deleted phases.

4. Check to see if there are members in the library which you do not need any longer. Delete these members with the Librarian command ACCESS SUBLIB=lib.sublib / DELETE mn.mt . For sublibraries with space reusage attribute AUTOMATIC, the space of the deleted members may show up under DELAYED SPACE/DELAYED LB's. Free it as described above.

If the above actions fail to provide enough free space, then you have to enlarge library space.

Perform the following steps dependent on the Librarian message:

```
L278 - VSAM data space exhausted when attempting to extend library
       IDCAMS: DEFINE SPACE       increase VSAM data space
               ALTER-ADDVOLUMES   add other volumes to VSAM data space

L268 - Maximum number of extents (16) allocated - no further extension
       possible

L201 - Library is full:
Libraries in VSAM managed space: DEFINE CLUSTER with larger
               primary/secondary allocation values
Libraries in BAM controlled space: Submit new DLBL/EXTENT
               statements
```

The execution of the LIBR step 'DELETE library' is strongly recommended. If performs the following actions:

- It verifies that the library is not in use.
- It destroys the library descriptor, thus making the extent inaccessible to Librarian services.

Scattered Library

Member data may get scattered under the following conditions:

1. All library space is used once. The Librarian does not start reusing library space until all space of all existing extents is used once. You can find out if a library is in space reusage mode by running the Librarian command TEST LIB=lib AREA=SPACE and checking the output line HIGH END REACHED under FREE SPACE DESCRIPTOR (YES means space is being reused).

   So if the free library space is already scattered member data may be scattered after catalog.

2. Two (or more) members are cataloged in parallel into a library and the number of output buffers is too small to hold the entire member. This will trigger several library space requests to catalog a member and the library space

requests for one member may be intermixed with the library space requests for other members, thus causing fragmentation.

The number of output buffers allocated by the Librarian is dependent on the partition size.

The free space inventory contains a pointer to free library blocks to be used to fulfill the next library space request. When a library is in space reusage mode, this pointer is set to the beginning of the member deleted last.

To get an overview of the free and used library blocks, execute the Librarian command TEST LIB=lib AREA=SPACE. This command displays (among others) the free space inventory of a library on the format of a bit map: a '1' indicates that the library block is used, a '0' that it is free. By looking at the output line 'HIGH WATER MARK: POSITION' you can find out at which position the next member will be stored. Thus, by looking at the sequences of contiguous '0' and by knowing the sizes of the members to be cataloged next, you can roughly predict whether these members will be scattered heavily, slightly or not at all. On a pure command basis you cannot store a member in a specific piece of free space.

COPY or BACKUP/RESTORE of a whole library does a re-organization. But be aware that a copied library may be bigger than the original one, because for a new index there will be always done a space reservation.

Does Member Scattering Impact Performance?

Generally, you can expect that scattering member data will not impact performance noticeably. To substantiate this answer, we have to look at specifics of the library structure and discuss the effect of scattering on different activities.

With respect to performance, retrieval of members is considered more critical than cataloging members. Therefore the library structure is designed to allow minimization of read-SIOs via contiguity counts:

- The number of contiguous library blocks starting from the beginning of the member is stored in the member directory entry.
- The control field of each library block contains the number of library blocks contiguously following this library block.

Members stored scattered in a library require more SIOs for retrieval. However, the above described contiguity counts allow building CCW-chains to read all contiguous blocks of a scattered member with a single SIO.

With this in mind, we may have a look at the overall system performance and system throughput. From a library point of view, the overall system performance is influenced mainly by the time it takes to fetch programs, that is, to load phases into partitions. Thus, the number of read-SIOs is the predominant factor. To notice the impact of scattering on the system performance, that is, the increase of read-SIOs, lots of heavily scattered phases must be fetched constantly which is not likely to happen.

Looking at Librarian functions, scattering is expected to cause a noticeable performance decrease only when masses of scattered members are processed.

Influences of BACKUP/RESTORE on Used Space

When a new sublibrary is filled with members (e.g. through RESTORE), the library blocks of the member index are filled only to about 50%. When members are added later on, the index library blocks may be completely filled without the need to split library blocks.

If you backup a sublibrary whose index library blocks are filled above 50% and restore this sublibrary, the LISTDIR command will show a higher number of USED SPACE/USED LB'S after RESTORE than it showed before RESTORE due to the new 50% utilization of the index library blocks. Since RESTORE LIB is the sum of restoring its sublibraries, these explanations apply also to RESTORE LIB.

When a library or sublibrary is backed up, its current number of used library blocks is recorded on the backup tape. To compensate for the 50% utilization of the index library blocks after RESTORE, the command RESTORE LIB SCAN as well as RESTORE SUBLIB SCAN displays size values which are increased by 10 library blocks. Thus, RESTORE SCAN will display library or sublibrary sizes which are slightly different from the actual ones.

Before the Librarian starts to restore a library into BAM controlled space, it checks to see if the library on the backup tape will fit into the disk extents. Again, the library size value recorded on the backup tape is increased by 10 library blocks before the check is made. No size check is made when a library is to be restored into VSAM managed space.

Restoring a sublibrary (RESTORE SUBLIB) starts without checking if the sublibrary on the backup tape will fit into the free space of the library (either in BAM controlled space or in VSAM managed space).

Cancel of Librarian Actions

Entering the commands

```
CANCEL xx       (z/VSE)
/CANCEL         (VSE/ICCF)
PCANCEL, PFLUSH (VSE/POWER)
```

does not bring the Librarian to an immediate stop. Instead, the program continues to run until either the library is in a consistent state (when updating) or the next part of listing is completed (when displaying library contents).

If you cancel when working with the Librarian in prompting mode either from SYSLOG or from a VSE/ICCF terminal, only the Librarian command in process is terminated, not the entire Librarian session. The Librarian terminates the command in process (with proper cleanup of the library) and prompts for the next command to be entered. Cancelling when the Librarian is waiting for the next command to be entered terminates the entire Librarian session.

Cancelling the Librarian running in batch mode terminates the job step, not just the command in process.

Entering the z/VSE command CANCEL xx,FORCE stops the partition immediately without giving the Librarian a chance to do cleanup. Therefore, when cancelled with FORCE during update, the library in process may be in an inconsistent state afterwards, e.g.:

- Library blocks may erroneously be flagged as taken, thus not available for further usage.
- The number of members displayed with LISTDIR SUBLIB may not be identical to the actual number of members in the sublibrary.
- The member index may not be updated properly, causing additional SIOs to locate a member.

To find out if a library is in an inconsistent state, use the Librarian command TEST. Even if a library is in an inconsistent state, you are often able to read the members in that library and, thus, recover through BACKUP/RESTORE commands.

If the system breaks down during update of a library, the potential effects are the same as with CANCEL xx,FORCE.

TRACE

TEST TRACE=LEVEL2|SIO|ALL

TEST TRACE=LEVEL2|SIO|ALL causes a message for every executed module in the specified function range. This may be used to determine accessed libraries/members (in this case TEST TRACE=LEVEL2 is sufficient).

Important Librarian Level 2 Functions

LBRACCESS handles LIBDEFS/ACCESS/CONNECT

- MF=GET, gets sublibrary from chain
- MF=ADD, puts sublibrary into chain
- MF=REMOVE, removes sublibrary from chain

INLMLCON/INLMSCON connects to a library/sublibrary

INLMMCON connects to a member

INLMFIND searches and connects to a member

INLMBLDL searches a list of members/sublibraries

INLMGETR reads a member record

INLMPUTR writes a member record

INLMDIS disconnects from a member

INLMSTOW handles directory entries

- MF=ADD, writes directory entries
- MF=DEL, removes directory entries
- MF=REN, replaces directory entries
- MF=PURGE, purges member data

Chapter 8. VSE/VSAM

Introduction

VSAM is the file management system for z/VSE. It is used to process files that are stored on DASD devices. VSAM is designed to meet the processing needs of demanding processing environments with high performance, ease-of-use, device independence, reliability, and recoverability. VSAM has been in use by numerous customers over many years and is the cornerstone to providing the 24-hour operation that z/VSE customers require. In the effort to meet all the requirements placed on it, VSAM can sometimes appear to behave in an unexpected manner. The following sections are the results of several years of experience with VSAM service and are intended to provide some insight into some potential VSAM problems. The capabilities and limitations of VSAM are discussed as well as suggestions for VSAM setup. Typical causes and solutions of VSAM open errors are described and migration pitfalls are also identified.

Note - for information on

- SAM ESDS support, see pp.67-69
- VSE/VSAM utilities, see pp.72-74

of the presentation "Design & Tuning of VSE/VSAM" from the *"2002 z/VM, VSE and Linux on IBM zSeries Technical Conference", Oct. 2002* , available at the location:
ftp://public.dhe.ibm.com/eserver/zseries/zos/vse/pdf3/techconf2002/E54.PDF

Capabilities and Limitations

VSAM File Size Limitations

There are several limiting factors in VSAM (number of tracks per extent, number of total extents). As quickly as we announce hardware that exceeds these limits, we release new VSAM code which supports the hardware, and extends the same limits.

There are really only six limiters on VSAM file size.

- You are allowed a maximum of 123 volumes for your file. However, with today's large volumes, you can very quickly get some very large files. For implicitly defined files, VSE/VSAM will extract up to 16 volumes from the default model
- Another limiter is the 4 byte RBA (Relative Byte Address). Every record in a VSAM file is ultimately accessed using its relative byte position from the beginning of the file. This means that you can only have 4.3 billion (4,294,967,295) bytes of data in a single VSAM file. This amounts to about 1/2 3390 model 9 dedicated to a single VSAM file. Since VSAM file compression occurs prior to the insertion of records into the control interval, usage of compressed files (introduced in VSE/ESA 2.1), substantially increases the

amount of user data that can be “packed” into 4GB. The actual limit is 1 control area less than 4GB.

With VSE/ESA 2.3, a cluster can be defined as 'ExtraLargeDataset' or 'XXL', which allows up to 289GB for an extended KSDS dataset.

- VSAM files are limited to 123 extents. This is normally only a problem if you specify a very small secondary extent relative to the primary extent. Remember, VSAM will sub-allocate an extent up to 5 times. So if you ask for 500 cylinders as a primary extent, and VSAM cannot find that much contiguous space, it will attempt to give the file up to 5 extents of 100 cylinders each (or some combination thereof). This may quickly exceed the extent limit for the file. Unique files and reusable files are limited to 16 extents.
- The lock used by Shareoption(4) processing has a two byte field for the control area id (or index control interval id for KSDS files). Thus, a Shareoption(4) file may not have more than 64511 (64K-1024) control areas, and the maximum file size would be 64511 times your control area size.

  If you use the maximum control area size for your file of one cylinder, define the file as XXL, and reside on 3380, with a CISIZE of 4K, this would set a maximum file size of 36.9GB.
- In addition, any single allocation (primary or secondary) cannot contain more than 16Meg records, if the user intends on using Backup/Restore. See APARs DY36234 and DY40943.
- When defining files using RECORDS(..), you can only specify up to 16 million records per extent. You can request more data by specifying a larger maximum recordsize for the file. Remember, if the file is compressed, VSAM uses the un-compressed maximum record length to calculate how much space to reserve for the file. This may give you more space than you actually need.
- With z/VSE 4.1, a VSAM catalog can be defined with a primary / secondary allocation of up to 4999 cylinders or 8.288.095 blocks.

Actually, customers will start running into severe performance degradation before they reach the outer limit of VSAM file size (merely from the number of index levels you have to run through to find a single record).

Setup

Reorganize a File

Since IDCAMS BACKUP/RESTORE processes files at the control interval level, it cannot be used to reorganize a file. You must use REPRO, or some utility that processes the file at the record level. It is important that the file be deleted and re-defined. Some utilities do this automatically.

Catalog Maintenance

Note - for additional information on

- warning concerning the use of VSE/Fast Copy to migrate catalog data see p.49
- backup/restore see pp.40-44, or
- local shared resource (LSR) under CICS see pp.57-69

of the presentation "Design & Tuning of VSE/VSAM" from the *"2002 z/VM, VSE and Linux on IBM zSeries Technical Conference", Oct. 2002* , available at the location:
ftp://ftp.software.ibm.com/eserver/zseries/zos/vse/pdf3/techconf2002/E54.PDF

Check a LISTCAT to see where the catalog growth is occurring. If it is mostly in the high-key range, you can re-build the catalog by simply REPRO'ing it out and back in (do not delete or re-define the catalog). Otherwise, if the catalog is corrupted, the catalog must be rebuilt by backing up all files, deleting and re-defining the catalog, spaces and files. The catalog cannot extend onto a second volume, so if the primary catalog volume fills up, you must also rebuild the catalog.

If you are restoring files into a new catalog using the VOLUME parameter, make the catalog volume the last volume in the list. This ensures that the catalog has space to extend into. Filling up the catalog volume may prematurely terminate the restore if the catalog has to extend.

Reorganize a Catalog

It is possible to enlarge a catalog using REPRO without backing up all files and deleting/redefining the catalog. These steps can also be used if a catalog is suffering from “wasteland” (see topic later in this chapter.) This can be a major time saver when a catalog owns a large number of files on many volumes. The indicated job streams are found in *VSE/VSAM Commands.*

1. Use the following jobstream to copy catalog records to tape.

```
// JOB    EXAMPL16
// ASSGN  SYS005,cuu       <===== Empty tape for backup
// MTC    REW,SYS005
// DLBL   IJSYSUC,'catalog filename',,VSAM
// TLBL   CATOUT,'PORTABLE.TAPE1',,TAPE01,1
// EXEC   IDCAMS,SIZE=AUTO
    REPRO INFILE(IJSYSUC/UCATMRPW)                      -
          OUTFILE (CATOUT                               -
            ENVIRONMENT (PDEV(2400) RECFM(VARBLK) REW   -
                         BLKSZ(5164) RECSZ(516)))
/*
/&
```

Figure 74. Copy Catalog Records to Tape Sample Job

Follow steps 2 - 4 only if you are enlarging the catalog. Otherwise, if you are simply reorganizing the catalog in its previous extents, go to step 5.

2. Remove all files with allocations on the catalog volume. This can be done by backing them up, or moving them to another volume. This is important. When we define the catalog in a larger extent, we must, of necessity delete all files from the catalog volume. This is detected during restore by REPRO, who then marks all files residing on the catalog volume as “NOTUSEABLE”. If you are simply reorganizing the catalog in its previous extents, this step is not required.
3. Scratch the VSAM spaces belonging to the user catalog from the catalog volume. If all space was owned by the user catalog, it is easier to scratch the entire catalog pack. Use a tool such as IKQVDU. Do not modify any of the other volumes belonging to this catalog.

4. Define a new, larger, user catalog on the original volume. Remember to define the CYL(..) parameter at the data and index level of the DEFINE UCAT. Space definitions at the cluster level only allocate general-use VSAM space, and the catalog is defined using a default of less than 10 tracks or 5 cylinders on BIG- and FAT-DASD.
5. Restore the old user catalog from tape using the following:

```
// JOB     EXAMPL17
// ASSGN   SYS004,cuu    <=== Tape containing previous backup
// MTC     REW,SYS004
// DLBL    IJSYSUC,'D27UCAT2',,VSAM
// TLBL    CATIN,'PORTABLE.TAPE1',,TAPE01,1
// EXEC    IDCAMS,SIZE=AUTO
     REPRO   INFILE (CATIN                            -
               ENVIRONMENT (PDEV(2400) RECFM(VARBLK)  -
                            BLKSZ(5164) RECSZ(516)))  -
             OUTFILE(IJSYSUC/UCATMRPW)
/*
/&
```

Figure 75. Restore User Catalog from Tape Sample Job

Re-Build a Catalog

If the catalog is corrupted, usage of the previous steps to re-organize the catalog will only copy the bad records into the new catalog. In order to correct the situation, all files in the catalog must be backed-up, the catalog and all controlled space deleted, re-defined, and all files restored. This re-builds all catalog information, including the corrupted pointers.

Please follow these steps:

1. Backup all files belonging to this catalog.

   We recommend using BACKUP (*) IDCAMS command, but you can use REPRO, EXPORT, or several generally available OEM packages. Remember, VSAM Backup/Restore does not backup VSE libraries (use LIBR for that), nor does it backup empty files (including default models). Since some types of DL/I files consist of multiple inter-related VSAM files, the applicable DL/I utilities should be used to backup DL/I files.
2. Identify all volumes which contain space from this catalog.

   This may be done with a LISTCAT SPACE IDCAMS job. It is important that no other VSAM catalog owns space on any of the affected volumes. If you have volumes on which multiple VSAM catalogs own space, you will have to identify which extents on the volume belong to this catalog by comparing the extent information. Otherwise, you can delete all VSAM spaces on the volume (see step 4).
3. If the catalog to be re-built is a user catalog, you will have to remove the pointer to the user catalog from the master catalog using an EXPORT DISCONNECT IDCAMS job. If you are re-building the master catalog, skip this step. However, you will have to re-IPL VSE in basic mode, since the PRD1 and PRD2 libraries (CICS, optional products, etc) are under VSAM control in the master catalog.

4. Delete all VSAM spaces on the affected volumes.

   If the volumes only contain VSAM spaces, you may use the following job.

```
// ASSGN SYS000,X'cuu'
// UPSI 11
// EXEC IKQVDU,SIZE=AUTO
```

Figure 76. Delete VSAM Space With IKQVDU Sample Job

   If the volume to be scratched also contains non-VSAM files, you must identify the VSAM file names from a VTOC listing. VSAM file names have the following format.

```
Z999999x.VSAMDSPC.Tnnnnnnn.Tnnnnnnn

      where x = '2' for a normal data space
                '4' for a space containing a user catalog
                '6' for a space containing the master catalog
            nnnnnnn = Time-of-day stamp when dataspace created.
```

Figure 77. VSAM File Name Format

   Use the following job to delete data spaces.

```
// ASSGN SYS000,X'cuu'
// EXEC IKQVDU,SIZE=AUTO
    --- Enter following in response to operator prompt
   RESET OWNERSHIP   ==> Remove VSAM ownership flag
   RESET CRA         ==> Only required if recoverable catalog
   SCRATCH           ==> Remove a label from VTOC
     <data set name> ==> Name of file to be deleted (when prompted)
   END               ==> When all files deleted on this volume
```

Figure 78. Delete VSAM Data Spaces Sample Job

   See “Maintaining VTOC and VOL1 Labels on DASD (IKQVDU)” in *VSE/VSAM User's Guide and Application Programming.*

5. After completely deleting all VSAM space and the affected catalog, if the catalog is a user catalog the catalog must be deleted from the master catalog. You do this using the IDCAMS command EXPORT DISCONNECT.

6. Now the user catalog must be redefined along with all data spaces.

   Redefine all VSE libraries if you backed any up using LIBR. Also redefine any required empty files (including default models).

7. Restore all files from the backup tapes created in step 1.

8. If you are re-building the master catalog, you must reconnect all user catalogs in the system now. Use IDCAMS IMPORT CONNECT for this.

   Note: If you are re-building a master catalog (or planning a migration of DOSRES or SYSWK1 between un-like devices), it is simpler to re-install z/VSE. The main problem is that you cannot enter IDCAMS commands (delete / re-define catalogs, spaces, clusters, backup / restore) from the console. So, in order to re-define the master catalog you have to be either running under VM, or have the jobs already prepared in the POWER reader. Remember, without

the VSAM master catalog, you cannot bring up CICS and the interactive interface.

Hint: An interactive tool named 'IDCONS.PHASE' is distributed in the z/VSE system library. IDCONS can be invoked from a 'PAUSE' job and used to perform VSE/VSAM commands, e.g. DEFINE MASTERCATALOG, RESTORE, etc.

Dataset Name Sharing

Note - for information on

- example of defining DSN using RDO, see p.63 (for example, concerning alternate indices and base cluster updates), see p.60
- further dataset name sharing information than discussed here, see pp.63-64

of the presentation "Design & Tuning of VSE/VSAM" from the *"2002 z/VM, VSE and Linux on IBM zSeries Technical Conference", Oct. 2002* , available at the location:
ftp://ftp.software.ibm.com/eserver/zseries/zos/vse/pdf3/techconf2002/E54.PDF

Dataset Name Sharing allows multiple opens of the same file to use the same in-core VSAM control block structure (including buffers). It is not required that the ACBs all have the same ddname. VSAM automatically detects that the ACBs refer to the same VSAM cluster, and adds the new ACB to the existing cluster, if requested. DSN is requested by coding ACB MACRF=(DSN,...) on the first, and all subsequent ACBs to be shared. Under CICS, DSNSHR=(ALL | UPDATE), BASE=<NAME> can be coded as an option to DFHFCT TYPE=FILE, or DSNSHARING under RDO (Resource Definition On-line) using CEDA.

Usage of dataset name sharing not only saves storage, but also prevents many variations of read integrity problems (see “Read Integrity Problems” on page 204) and allows read/write sharing of SHR(2) files (e.g. through alternate indexes). However, there are some pitfalls of which you should be aware. Programs accessing VM/VSAM using the MVS versions of COBOL and PL/1 automatically turn on dataset name sharing, with several undesirable results.

1. Return Code x‘A2’ (Opposite Input or Output mode)

   The first open of a shared ACB determines the type of structure to be built (read/only or read/write). If the first ACB to be opened requests only read/only access, a read/only structure is built and all further ACBs to share this structure must be also for read/only. The same applies for read/write. If the first ACB requests read/write access, an attempt to open a read/only ACB against this structure will result in OPEN return code x‘A2’.

   Note: CICS automatically adjusts the SERVREQ definitions for all files defined in the FCT as “Data-set Name Sharing”.

2. Return Code x‘A3’ (Insufficient strings BSTRNO)

   The total number of strings (simultaneous accesses) for a shared structure must be specified in the initial open. A new ACB operand (BSTRNO) was added to allow specification of additional strings which would be available for sharing ACBs. This applies to alternate indexes as well, so opening a base cluster with five alternate indexes requires 6 strings. CICS will automatically calculate the number of strings using the name in the “BASE” operand. However, there is no way to specify additional strings in a COBOL or PL/1 program. In order to address this, VSAM automatically doubles the number of

strings specified in the ACB, but ensures that it adds at least 5 additional. Thus, since there is no provision for specification of the number of strings in a PL/I or COBOL file structure definition, you are limited to 6 ACBs which share the same structure.

Open Errors

Return Code 168 x‘A8’

Example message.

```
F4 004 4228I FILE KSDS       OPEN  ERROR X'A8' (168) CAT=UCAT
(OCSHR--5) FILE ALREADY OPEN IN ANOTHER PARTITION, RC X'04' TASK X'0020'
```

Following errors will result in return code x‘A8’ during OPEN.

1. Attempt to OPEN a non-SHR(4) file while the same file is opened in another partition or same partition. Extended message will tell customer is file is open in this partition or another partition. Remember that OPENs over a path (aix or base) count, unless the customer is using DataSet Namesharing.
2. KSDS files can be opened in one of three modes: Keyed, Addressed, or CI-mode. Once a SHR(4) file is opened in one of these modes, all other OPENs must be in the same mode. (OCSHR--2). This often happens sharing files between CICS partitions, particularly with the IESCNTL file. IESCNTL must be specified as SHR(4) if it is to be shared between CICS partitions.
3. If the file is shared between multiple systems, ensure that the CPUID has not changed. The external locks include the CPUID in the key, and if the CPUID is changed, old locks under the old CPUID are not cleared. If VSE systems are run second level under VM, changing the CPUID is often done using the CP SET CPUID command, which will drive a different ASI proc. This may result in ghost lock entries being left in the file, which are not freed. Use the AR UNLOCK command to release lock entries for the bogus CPUIDs.
4. If the file is shared between multiple releases of VSE, check to ensure that DOSRES or SYSWK1 are not marked as ,SHR on either system. Even though each system has their own version of DOSRES or SYSWK1, or only link to the appropriate volume, all locks are done using only the volume of the catalog and the CI number within the catalog. Thus, if two catalogs are defined on DASD with a duplicate VOLID, it is easy to pick up a lock from one release / catalog, and find that it matches the CI from the other release / catalog, and reject the OPEN. Therefore unique VOLIDs are essential in a shared environment.
5. When a file is being initially loaded (prior to first CLOSE), the SHAREOPTIONS for the file are temporarily reset to ‘1’. This means, even if the SHAREOPTIONS are ‘2’ or ‘4’, another partition will get RC x‘A8’ attempting to open the file, even for input.
6. Close of a file under CICS may not always be successful. This is the case whether the close is issued via CEMT command, or from one of a number of available vendor products which issue a request from batch to CICS to close an on-line file so it can be opened for output in batch. CICS will not close the file if there is an outstanding request against the file. This includes an update request that has not been either returned or released to File Control.

7. When multiple files are defined to CICS FCT as “Dataset Name Sharing”, CICS automatically adjusts the SERVREQ of all files to output. Thus, even if you close the FCT entry which has the file opened for output, other FCT entries defined only for input may be sharing the same structure and may have been opened for output.

With z/VSE 4.2, the A8 return code and message was changed. In case a lock is held by another partition on the same VSE system, then the corresponding task id of the task holding the lock will be presented within in the A8 message. In case the lock is held by a task running in a partition on a different VSE system, then the task id X'FFFF' is presented instead.

As of VSE/ESA 1.3, a new lock trace option has been added to VSE. This will display, on the console, the VSE supervisor lock control blocks, and can be used to identify the partition owning a lock. Lockname may be generic, that is, “V*” will trace all VSAM locks.

See a detailed description of this command earlier in this document under *Lock Manager Service Enhancements*.

Using Lock Trace to identify owner of VSE/VSAM lock

```
LOCK TRACE=<partition>,V*
  -then-
LOCK TRACE=OFF
```

1. If the failing job is being run in a dynamic partition, remember that the partition control blocks do not exist until the job is actually run. Put a // PAUSE in the failing job and enter the “LOCK TRACE=<partition>” command at that point.
2. Locktab entries and Owner elements are matched pairs.
3. If the error produces Message 4228I rcx'A8', then it is easy. The Locktab and Owner entry are immediately prior to the Message in the console log.

Otherwise ...

4. Scan the trace entries ignoring all locktab entries which do not contain “V” plus the volume name of the user catalog where the file in question resides.
5. Ignore all locktab entries in the following format: “V”<volser>nnnnmmmm, where nnnn or mmmm = ‘0000’ or ‘0001’
6. We are looking for a locktab entry in the following format: “V”<volser>iiiiii0000. iiiiii will be the control interval number of the file in question within the user catalog. In the example, x‘E5C3E3E2 F2F2F000 005A0000’ (‘VCTS220..|..’). The user catalog resides on volume CTS220, and the file in question is ci# x‘00005A’.
7. In some cases, there will be two locktab entries following each other, that match this format. This would be for a KSDS file, and the first one describes the data component, and the second, the index.
8. Having located the failing locktab entry, move to the Owner element immediately following. +5 (counting from zero) contains the task id of the owner of the file in question. x‘21’ = BG, x‘22’ = F1, x‘23’ = F2 ... x‘2B’ = FA, x‘2C’ = FB, x‘30’ - x‘99’ refer to either a VSE sub-task or dynamic partition. Use the AR “STATUS” command to identify the partition / sub-task.

In the following example, the file in question is owned by BG (task id = x‘21’).

```
F4 0004 // JOB LOADKSDS
        DATE 02/08/1999, CLOCK 21/41/38
F4 0004 READY TO OPEN KSDS FILE

F4 0025 LOCKTAB ENTRY
V0004EA18  02CD2EA0 00000000 E5E2E8E2 D4C3D600 * ò ff    VSYSMCO *  R0004EA18
V0004EA28  00000000 11800001 0004EA38 00000000 *     Ø    ⅝     *  R0004EA28
F4 0025 OWNER ELEMENT
V02CD2EA0  00000000 00250000 00011000 00000000 *                 *  R00975EA0
F4 0025 LOCKTAB ENTRY
V0004EA78  02CD2BB0 00000000 E5C3E3E2 F2F2F000 * ò ¬    VCTS220 *  R0004EA78
V0004EA88  00000001 04C00000 02CD2FE0 0004EA58 *     ä   ò Ö  ⅝ì*  R0004EA88
F4 0025 OWNER ELEMENT
V02CD2BB0  00000000 00210001 00000000 00000000 *                 *  R00975BB0
F4 0025 LOCKTAB ENTRY
V0004EA18  02CD2EA0 00000000 E5E2E8E2 D4C3D600 * ò ff    VSYSMCO *  R0004EA18
V0004EA28  00000000 11800001 0004EA38 00000000 *     Ø    ⅝     *  R0004EA28
F4 0025 OWNER ELEMENT
V02CD2EA0  00000000 00250000 00011000 00000000 *                 *  R00975EA0
F4 0025 LOCKTAB ENTRY
V0004EA18  02CD2EA0 00000000 E5E2E8E2 D4C3D600 * ò ff    VSYSMCO *  R0004EA18
V0004EA28  00000000 11800001 0004EA38 00000000 *     Ø    ⅝     *  R0004EA28
F4 0025 OWNER ELEMENT
V02CD2EA0  00000000 00250000 00011000 00000000 *                 *  R00975EA0
F4 0025 LOCKTAB ENTRY
V02CD2FC0  02CD2BC0 00000000 E5C3E3E2 F2F2F000 * ò ä    VCTS220 *  R00975FC0
V02CD2FD0  00010001 04C00000 02CD2FA0 02CD2FE0 *     ä   ò ff ò Ö*  R00975FD0
F4 0025 OWNER ELEMENT
V02CD2BC0  00000000 00210001 00000000 00000000 *                 *  R00975BC0
F4 0025 LOCKTAB ENTRY
V0004EA18  02CD2EA0 00000000 E5E2E8E2 D4C3D600 * ò ff    VSYSMCO *  R0004EA18
V0004EA28  00000000 11800001 0004EA38 00000000 *     Ø    ⅝     *  R0004EA28
F4 0025 OWNER ELEMENT
V02CD2EA0  00000000 00250000 00011000 00000000 *                 *  R00975EA0

F4 0025 LOCKTAB ENTRY
V02CD2AE0  02CD2B10 00000000 E5C3E3E2 F2F2F000 * ò      V CTS220  *  R00975AE0
V02CD2AF0   005A 0001 04C00000 02CD2AC0 02CD2B20 * |   ä   ò ä ò  *  R00975AF0
F4 0025 OWNER ELEMENT
V02CD2B10  00000000  0021 0001 00000000 00000000 *                *  R00975B10
F4 0025 LOCKTAB ENTRY
V02CD2AE0  02CD2B10 00000000 E5C3E3E2 F2F2F000 * ò      VCTS220 *  R00975AE0
V02CD2AF0  005B0001 04C00000 02CD2AC0 02CD2B20 * |   ä   ò ä ò  *  R00975AF0
F4 0025 OWNER ELEMENT
V02CD2B10  00000000 00210001 00000000 00000000 *                 *  R00975B10

F4 0004 4228I FILE KSDS       OPEN  ERROR X'A8'(168) CAT=IJSYSCT
   (OPNAB-15) DATASET ALREADY BEING LOADED BY ANOTHER ACB

F4 0004  RETURN CODE 008 (X"A8") FROM OPEN     DISPLACEMENT: 0000CE

F4 0025 LOCKTAB ENTRY
V0004EA78  02CD2BB0 00000000 E5C3E3E2 F2F2F000 * ò ¬    VCTS220 *  R0004EA78
V0004EA88  00000001 04C00000 02CD2FE0 0004EA58 *     ä   ò Ö  ⅝ì*  R0004EA88
F4 0025 OWNER ELEMENT
V02CD2BB0  00000000 00210001 00000000 00000000 *                 *  R00975BB0
```

Figure 79 (Part 1 of 2). Example Where Message 4228I Is Issued

```
F4 0025 LOCKTAB ENTRY
V02CD2FC0  02CD2BC0 00000000 E5C3E3E2 F2F2F000 * ò ä    VCTS220 *  R00975FC0
V02CD2FD0  00010001 04C00000 02CD2FA0 02CD2FE0 *     ä   ò ff ò Ö*  R00975FD0
F4 0025 OWNER ELEMENT
V02CD2BC0  00000000 00210001 00000000 00000000 *               *  R00975BC0
F4 0004 1S78I  JOB TERMINATED DUE TO  PROGRAM ABEND
F4 0004 EOJ LOADKSDS
        DATE 02/08/1999, CLOCK 21/41/42, DURATION   00/00/03
```

Figure 79 (Part 2 of 2). Example Where Message 4228I Is Issued

Where:

- **CTS220** indicates the Catalog volume
- **005A** indicates the Cluster CI#
- **0021** indicates that partition (BG) has this cluster open

```
    DELETE (TEST.KSDS.CLUSTER) CLUSTER PURGE -
       CATALOG(VSAM.TEST.CATALOG)
IDC3028I DATA SET IN USE
IDC3009I ** VSAM CATALOG RETURN CODE IS 184 - REASON CODE IS IGG0CLCX-4
IDC0551I **ENTRY TEST.KSDS.CLUSTER NOT DELETED
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 8

F4 0004 // JOB DELETE CLUSTER
        DATE 02/08/1999, CLOCK 20/49/05
F4 0025 LOCKTAB ENTRY
V0004EA18  0004EAC8 00000000 E5E2E8E2 D4C3D600 *  §H    VSYSMCO *  R0004EA18
V0004EA28  00000000 11800001 0004EA38 00000000 *     Ø    §     *  R0004EA28
F4 0025 OWNER ELEMENT
V0004EAC8  00000000 00250000 00011000 00000000 *               *  R0004EAC8
F4 0025 LOCKTAB ENTRY
V0004EA78  02CD2BB0 00000000 E5C3E3E2 F2F2F000 * ò ¬    VCTS220 *  R0004EA78
V0004EA88  00000001 04C00000 02CD2FE0 0004EA58 *     ä   ò Ö  §ì*  R0004EA88
F4 0025 OWNER ELEMENT
V02CD2BB0  00000000 00210001 00000000 00000000 *               *  R00975BB0
F4 0025 LOCKTAB ENTRY
V0004EA18  0004EAC8 00000000 E5E2E8E2 D4C3D600 *  §H    VSYSMCO *  R0004EA18
V0004EA28  00000000 11800001 0004EA38 00000000 *     Ø    §     *  R0004EA28
F4 0025 OWNER ELEMENT
V0004EAC8  00000000 00250000 00011000 00000000 *               *  R0004EAC8
F4 0025 LOCKTAB ENTRY
V0004EA18  0004EAC8 00000000 E5E2E8E2 D4C3D600 *  §H    VSYSMCO *  R0004EA18
V0004EA28  00000000 11800001 0004EA38 00000000 *     Ø    §     *  R0004EA28
F4 0025 OWNER ELEMENT
V0004EAC8  00000000 00250000 00011000 00000000 *               *  R0004EAC8
F4 0025 LOCKTAB ENTRY
V02CD2FC0  02CD2BC0 00000000 E5C3E3E2 F2F2F000 * ò ä    VCTS220 *  R00975FC0
V02CD2FD0  00010001 04C00000 02CD2FA0 02CD2FE0 *     ä   ò ff ò Ö*  R00975FD0
F4 0025 OWNER ELEMENT
V02CD2BC0  00000000 00210001 00000000 00000000 *               *  R00975BC0
F4 0025 LOCKTAB ENTRY
V0004EA18  0004EAC8 00000000 E5E2E8E2 D4C3D600 *  §H    VSYSMCO *  R0004EA18
V0004EA28  00000000 11800001 0004EA38 00000000 *     Ø    §     *  R0004EA28
```

Figure 80 (Part 1 of 2). Example Where Message 4228I Is Not Issued

```
F4 0025 OWNER ELEMENT
V0004EAC8  00000000 00250000 00011000 00000000 *               *  R0004EAC8

F4 0025 LOCKTAB ENTRY
V02CD2B80  02CD2B60 00000000 E5C3E3E2 F2F2F000 * ò -    V CTS220  *  R00975B80
V02CD2B90  005A 0000 14C00001 02CD2B40 02CD2BE0 * |   ä   ò   ò Ö*  R00975B90
F4 0025 OWNER ELEMENT
V02CD2B60  00000000  0021 0000 00011000 00000000 *                *  R00975B60
F4 0025 LOCKTAB ENTRY
V02CD2AC0  02CD2AA0 00000000 E5C3E3E2 F2F2F000 * ò ff    VCTS220 *  R00975AC0
V02CD2AD0  005B0000 14C00001 02CD2A80 02CD2AE0 * $   ä   ò Ø ò Ö*  R00975AD0
F4 0025 OWNER ELEMENT
V02CD2AA0  00000000 00210000 00011000 00000000 *               *  R00975AA0

F4 0025 LOCKTAB ENTRY
V0004EA78  02CD2BB0 00000000 E5C3E3E2 F2F2F000 * ò ¬    VCTS220 *  R0004EA78
V0004EA88  00000001 04C00000 02CD2FE0 0004EA58 *     ä   ò Ö  ‰ì*  R0004EA88
F4 0025 OWNER ELEMENT
V02CD2BB0  00000000 00210001 00000000 00000000 *               *  R00975BB0
F4 0025 LOCKTAB ENTRY
V02CD2FC0  02CD2BC0 00000000 E5C3E3E2 F2F2F000 * ò ä    VCTS220 *  R00975FC0
V02CD2FD0  00010001 04C00000 02CD2FA0 02CD2FE0 *     ä   ò ff ò Ö*  R00975FD0
F4 0025 OWNER ELEMENT
V02CD2BC0  00000000 00210001 00000000 00000000 *               *  R00975BC0
F4 0004 EOJ DELETE    MAX.RETURN CODE=0008
        DATE 02/08/1999, CLOCK 20/49/10, DURATION   00/00/05
```

Figure 80 (Part 2 of 2). Example Where Message 4228I Is Not Issued

Where:

- **CTS220** indicates the Catalog volume
- **005A** indicates the Cluster CI#
- **0021** indicates that partition (BG) has this cluster open

To check if this is the right lock entry, you can use the following job:

```
// JOB IDCAMS PRINT CATALOG CONTROL INTERVAL
// DLBL UCAT,' xxx.yyy.zzz ',,VSAM,CAT=UCAT
// EXEC IDCAMS,SIZE=AUTO
   PRINT INFILE(UCAT) SKIP(X' 005A ') COUNT(1)
/&
```

Figure 81. Job to print a catalog CI number

Where:

- fi **xxx.yyy.zzz** is the catalog name
- fi **005A** is the cluster CI number

```
   PRINT INFILE(UCAT) SKIP(X'005A') COUNT(1)

000000  0000 005A  01000000 00000000 00000000   *                *
000010  00000000 00000000 00000000 00000000   *                *
000020  00000000 00000000 00000000 C3008D00   *             D   *
000030  6CE3C5E2 E34BD2E2 C4E24BD5 F14BC4C1   *% TEST.KSDS.N1.DA *
000040  E3C14040 40404040 40404040 40404040   * TA               *
000050  40404040 40404040 40404040 40FFFFFF   *                *
000060  FFFFFFFF FF03150F 00000F00 00000000   *                *
000070  00030000 00020100 00060202 00000044   *                *
 ...
```

Figure 82. Print catalog CI number output

This indicates the lock occurred for file TEST.KSDS.N1.DATA.

Sometimes in the LOCK TRACE output, either the LOCKTAB ENTRY is empty or the OWNER ELEMENT is not displayed.

```
F4 0025 DTL
V00501F30  ........ ........ 001E1248 E5C3E3E2 *           « VCST *  R0E19EF30
V00501F40  F2F2F0 00 005A0000  C3E3E2F2 F2F00004 * 220   K  VCST220 *  R0E19EF
V00501F50  00000000 0000                        *                *  R0E19EF50
F4 0025 LOCKTAB ENTRY
V043D1DA0   00000000 00000000 00000000 00000000  *                *  R04267DA0
V043D1DB0   00000000 00000000  043D1D80 043D1DE0 *           ø   »*  R04267DB0

F4 0005 4228I FILE KSDS       OPEN  ERROR X'A8'(168) CAT=IJSYSCT
   (OPNH1-45) FILE ALREADY OPEN IN ANOTHER PARTITION
```

Figure 83. Example where locktab entry is empty

Generaly this indicates that the VSAM catalog is shared between several VSE systems and the lock is owned by another VSE system.

In the above example, the lock name is VCTS220 in character format followed by 00005A0000 in hexadecimal
On each VSE system sharing the catalog, type the command:
LOCK SHOW=VCTS220'00005A0000 (replace VCTS... with your lock name)
The quote (') is used to separate the character and hexadecimal strings.

```
LOCK SHOW=VCTS220'00005A0000
AR 0015 1I40I  READY
```

Figure 84. Example where a locktab entry is not found

If no locktab entry is displayed, repeat the command on the other VSE systems.

```
LOCK SHOW=VCTS220'00005A0000
AR 0025 LOCKTAB ENTRY
V044D2580  044D2490 00000000 E5E5E2C5 F3F0C100 * ( └     VCTS220  *  R00780580
V044D2590   005A 0000 12D00001 044D2500 044D25A0 * K   }   (   ( K*  R00780590
AR 0025 OWNER ELEMENT
V044D2490  00000000  0021 0000 00011000 00000000 *                *  R00780490
AR 0015 1I40I  READY
```

Figure 85. Example where a locktab entry is found

The owner element belongs to task 21 which is BG. This means the lock is owned by BG in the VSE system where the LOCK SHOW command is issued.

If the lock owner is still not found, you probably did not check all the active CPUs.

To check all active CPUs:

At a VSE console, type DLF and immediately press the PF7 key to display the IPL commands at the console, or print the $IPLxxx.PROC
($IPLxxx procedure name is displayed by the SIR command).
Locate the DLF command, example:

```
DLF UNIT=F09,BLK=4,NBLK=770,DSF=N,NCPU=6
```

and display the first record of the LOCK file:

```
DITTO/ESA FOR VSE              DB - DISK BROWSE
 PBN 0000004                                        COL 1        FORMAT HEX
VOLUME LOCKFL 0F09 9336
PBN LENGTH BYTE ---------------- HEX -----------------   ------ CHAR ------
4      512 0000 D3C60006  02000301  001C0012  00000004   * LF               *
           0010 00000190   8000FF11  11119672  8000FF66    *  0       0     *
           0020  66669672  8000FF77  77779672  8000FF55    *  0      *
           0030  55559672  8000FF44  44449672   00000000   *  0       0*
           0040 00000000  00000000  00000000  00000000   *                 *
           0050 00000000  00000000  00000000  00000000   *                 *
           ...
```

Figure 86. First record of the LOCK file

- fi LF is the lock file indicator
- fi CPU list starts at offset x'14'
- fi Each CPU entry is 8 bytes long: 8000FF1111119672, 8000FF6666669672, ...
- fi X'80' indicates that this CPU is active
- fi CPU number starts at offset 2: FF1111119672, FF6666669672, ...

Return Code 254 x‘FE’

IDCAMS RC254 documents a condition where the VSE/VSAM space map in the catalog does not match the VTOC (physical device characteristics). It is usually caused by a customer restoring (using Fast Copy, DDR, or a similar full-pack copy utility), a small disk (which is “owned” by a VSAM catalog) onto a larger drive. The problem is that VSAM catalogs maintain a “bit map” describing each volume owned by the catalog. The catalog also contains physical information relating to the DASD hardware characteristics (that is, type of DASD, number of tracks, cylinders, etc). Copying the volume using a full-pack copy utility does not change the internal description of the volume in the catalog, with results that can be predicted.

If the copy is between volumes with differing device characteristics, the results are usually immediate and disastrous. However, if the copy was between volumes with the same device characteristics, just a different number of cylinders, the problems don't normally show up until the customer attempts to define space in an area of the volume not covered by the catalog bitmap. IDCAMS DEFINE SPACE will detect this condition and issue a warning message with rc254.

OPEN Error X'FE' indicates a non-zero return code from the AF lock manager. The second line (extended message line) will list the AF lock manager return code. These are documented in the *z/VSE System Macros Reference.*

```
AF Lock Manager Return codes:

00   Successful
04   Resource Not Available (Lock queued if LOCK WAITECB)
08   Lock Table Full
0C   Inconsistent Request
10   A Deadlock Would Occur
14   DTL Format Error
18   Already Owned
1C   Lock File Full
20   Volume Not On-line
24   I/O Error On Lock File
```

An OPEN error 254 with LOCK return code x'0C' (Inconsistent Request) can be caused if the file resides on a shared volume with a non-unique VOLID, e.g. another z/VSE system added a disk as shared with the same VOLID. Even though each system has its own disk, all locks are done using the volume of the catalog and the CI number within the catalog. Thus, if two catalogs are defined on DASDs with a duplicate VOLID in two different z/VSE systems, a lock from one system may find a match from the other system, and reject the OPEN. Therefore unique VOLIDs are essential in a shared environment.

File Not Found

```
return code x'80':  DLBL not found
return code x'B4':  Cannot open catalog
return code x'94':  Cannot find file in catalog
```

When you receive one of the above messages, it is important to consider the source of the message before spending a lot of time and resource in problem correction.

Migration

Migrating Master Catalog

Please follow these steps to re-build your master catalog after migrating between un-like devices (that is, 3380 to 3390).

1. All packs containing non-VSAM files (e.g. DOSRES) should be copied using Fast Copy or some other full-pack copy tool.
2. Re-build all user catalogs using instructions in “Re-Build a Catalog” on page 184. Remember, before you EXPORT DISCONNECT any catalog (particularly the one on SYSWK1 = VSESPUC), check to ensure it does not contain any libraries needed by any running programs.
3. Don't forget to catalog any default models which belong in the catalogs you have re-built (especially DEFAULT.MODEL.ESDS.SAM).
4. Now you can proceed to re-build the master catalog. In order to do this, you have take down any programs which may need files / libraries resident in the master catalog (ICCF, CICS, VTAM, etc). The easiest way (if you are running VSE as a guest under VM), is to re-ipl, and terminate the IPL procedure as

soon as BG comes up, then run as a single partition. (See attached console log Figure 87 on page 196).

5. If you are running VSE as a guest under VM, you may punch the jobs from your CMS userid to the VSE machine using the following commands.

   ```
   CP SP D <vsemachine> CL A NOHO NOCONT
   PUNCH <job> (NOH
   ```

   Otherwise, you will have to prepare the required jobs ahead of time, and leave them in a power reader, then bring-up VSE with only POWER ("MINI" configuration). To do this, enter MSG BG instead of CANCEL BG,NODUMP, then enter MINI when prompted.

6. Perform the following steps.

   a. Delete VTOC entries for Master catalog using IKQVDU (Check ahead of time which extents are actually owned by the master catalog, because VSESPUC user catalog may also own extents on DOSRES).
   b. IDCAMS job to define a new master catalog.
   c. IDCAMS job to restore VSAM files back into master catalog.
   d. LIBR job to restore libraries back into master catalog.
   e. IDCAMS job to catalog new DEFAULT.MODEL.ESDS.SAM into mcat.
   f. IDCAMS job to IMPORT CONNECT the user catalogs into mcat.

7. You should now be ready to re-ipl the system.

```
BG 000 // JOB BGINIT

DATE 07/11/94,CLOCK 14/49/39
BG 000 1I93I RECORDER FILE IS   1% FULL
BG 000 IESI0221I PARTITIONS F3 F2 F1 WILL BE INITIALIZED IN RECOV START MODE.
BG 000 IESI0222I REMAINING PARTITIONS WILL BE INITIALIZED IN WARM START MODE.
       IF YOU WANT TO INTERRUPT THEN ENTER " MSG BG ".
cancel bg,nodump
CANCEL BG,NODUMP
AR 015 1I40I  READY
BG 000 0S01I THE OPERATOR CANCELED THE JOB
BG 000 0S00I JOB BGINIT   CANCELED.
BG 000 1N90I  EOP WAS FORCED BY EOJ
BG 000 1S78I  JOB TERMINATED DUE TO  CANCEL COMMAND
BG 000 EOJ BGINIT

DATE 07/11/94,CLOCK 14/49/43,DURATION   00/00/04
BG 000 1C10D  PLEASE ASSIGN SYSRDR.
BG-000
0 assgn sysin,00c
0 ASSGN SYSIN,00C
BG-000
0 assgn syspch,00d
0 ASSGN SYSPCH,00D
BG-000
0 assgn syslst,00e
0 ASSGN SYSLST,00E
BG-000
#cp sp c cont noho cl a
#cp sp d <userid>  nocont cl d
#cp sp e <userid>  nocont cl e
RDR FILE 0025 SENT FROM <userid> PUN WAS 0496 RECS 0003 CPY  001 A NOHOLD NOKEE
0
0
BG 000 // JOB DEFINE MASTER CATALOG

DATE 07/11/94,CLOCK 14/52/11

BG 000 EOJ DEFINE

DATE 07/11/94,CLOCK 14/52/13,DURATION   00/00/02
BG 000 1C00A  ATTN. 00C
BG-000
```

Figure 87. Re-Build VSAM Master Catalog Console Log

Migrating of Recoverable Catalogs

Beginning with z/VSE 3.1, the ability to define catalogs as recoverable was removed.

- Although existing recoverable catalogs were supported, new catalogs could no longer be defined as recoverable.
- The catalog-recovery functionality was no longer state-of-the-art and made support of newer devices increasingly difficult.

Beginning with z/VSE 4.3, the ability to extend files in a recoverable catalog or add additional space to a recoverable catalog was removed. The removal of the necessity to check for access to a recoverable catalog from all VSAM operations allows for more streamlined and better-performing code.

If you still use recoverable catalogs, you should plan to migrate these recoverable catalogs to standard catalogs. This should be done before you migrate to z/VSE 4.3 or to a later z/VSE version.

If you have already migrated to z/VSE 4.3 or to a later z/VSE version with existing recoverable catalogs, please contact IBM Support for assistance.

The migration steps are fairly straightforward and consist of:

1. IDCAMS BACKUP (*) a backup of all clusters from the catalog 2.
2. Depending on the volume layouts, you might have to identify which extents on a volume belong to this catalog by using extent information from an IDCAMS LISTCAT SPACE.
   - If the volume only contains VSAM space belonging to this catalog, you can use IKQVDU to scratch the volume that contains the catalog and all volumes containing its space.
   - Otherwise, refer to the topic "Re-Build a Catalog" in this manual.
3. IDCAMS EXPORT DISCONNECT to remove the user catalog entry from the master catalog.
4. IDCAMS DEFINE USERCATALOG.
5. IDCAMS DEFINE SPACE on all available volumes.
6. IDCAMS RESTORE OBJECTS(*) to restore all clusters to the catalog.

The removal of the support for recoverable catalogs will improve VSAM performance and ease the support of newer devices.

VSAM Migration Consideration

Migration of VSAM files using IXFP Flashcopy or VSE/Fast Copy are 2 ways of moving data in a vse environment. Although Flashcopy is much quicker then VSE/Fast Copy, they both subject to the following restriction. Failure to comply with these restrictions may result in the the VSAM catalog or data being corrupt or unusable.

1. All volumes for a particular catalog must be backed up at the same time.
2. If another catalog shares space on the same volume(s) as the catalog being migrated, that data will not be usable on the target volume, unless the other catalog is also backed up and included within the list of SOURCEVOLUMES used by Flashcopy.
3. All VSAM Clusters in the catalog(s) being backed up must be closed. Otherwise, you are exposed to data integrity issues, both because of data which may still be in incore buffers, and data which may be changed while the volumes are being copied.

For VSE/Fast Copy, the following additional restrictions apply:

1. Do not attempt to only copy specific cylinder / RBA ranges. The entire volume must be copied.
2. The source and target volumes of Fast Copy need to be the same type and size.

3. If Dump, Restore, Copy VOLUME is used, all files defined to the VTOC will be processed. The VTOC on the target volume must be at the same location as the source volume.
4. If Dump, Restore, Copy ALL is used, the volume is copied physically, without regard to the VTOC. Therefore, the location of the VTOC is not important, since it will be restored with the rest of the volume. For more information please see Chapter 10 of the *VSE/VSAM User's Guide and Application Programming* and for VSE/Fast Copy please see *System Utilities.*

Backup / Restore

General Notes

1. IDCAMS BACKUP is always done only at the physical record (Control Interval) level.
2. If the restore is being performed to exactly the same environment as the file had prior to backup (that is, device type, allocation, CISIZE then the restore will be performed in the “old” path (that is, without remap). In this format, only CAs will be restored (CA splits will be moved so that CAs are contiguous).
3. If anything changes, then the remap function will be invoked, and the restore will take place at the Control Interval level. This means that CIs will be moved to be contiguous. However, in no case does Backup or Restore actually manipulate individual records. Therefore, Backup/Restore cannot be used to reorganize a file at the record level.

Restore MSG IDC31340 (Backup File in Error)

If you experience msgIDC31340 “Backup File in Error”, consider the following.

1. Have you specified the STDLABEL parameter correctly in the IDCAMS RESTORE job?
2. Under some cases, DYNAM/T will cause this error. Please disconnect DYNAM/T and try the job again.
3. Did you attempt to stack multiple BACKUP passes on a single tape. This is not supported.

Backup File Integrity (Return Code 41 x‘29”)

VSAM provides a locking mechanism to avoid updates to files while they are being backed up. This locking mechanism will detect if a file to be backed up is open for output in another partition. In this case, it issues Message 4228I OPEN return code x‘29’, which is a warning message; the backup writes a message to SYSLST with the 44-character name of the file which is suspect, then continues with return code x‘04’.

Every attempt should be made to only backup files after they have been closed in all other partitions. Backup has been modified to lock the file during backup, as far as possible. If the file is not currently open for write access, Backup locks the file to prevent an application from opening it for output until backup for this file is complete. If the file is already opened for output, backup will proceed , but a warning message will be written to the console with additional information written to the backup log on SYSLST.

SYSLST file:

```
MSGIDC11310I <filename> MIGHT BE INCONSISTENT IN BACKUP
```

The following message is written to the system console:

```
4228I  FILE VDWACB  OPEN ERROR X'29' (041) CAT=<catname>
       (IKQOPNHC) WARNING : SHRx FILE ALREADY OPEN FOR OUTPUT DURING BACKUP!
```

Backup/Restore between ECKD and SCSI

With z/VSE 4.1, VSAM extended the Backup/Restore remapping functionality to allow an Extralargedataset (XXL) to be moved between ECKD and SCSI devices. This will enable customers to use Backup/Restore to move all types of VSAM clusters between ECKD and SCSI devices.

Backup/Restore Cross-Reference-Listing

With z/VSE 4.2, VSAM extended the Backup/Restore functionality by providing the new XREF and XREFONLY and NOXREF parameters.

- NOXREF specifies that the cross-reference listings will not be produced but object restoration will be performed.This is the default.
- XREF specifies that both the cross-reference listings will be produced and objects restoration will be performed.
- XREFONLY specifies that only the cross-reference listings will be produced and thus object restoration will not be performed.

Catalogs

Potential Corruption Causes

Although the probability of losing an entire catalog is very low, just as in all areas involving I/O in a computer system, the possibility of damage exits. As stated above, a VSE/VSAM catalog is itself a “file”. This means that anything that could cause a file corruption could theoretically cause a catalog corruption. This of course includes peremptory halting of some I/O process. This can include such things as a canceled task, a partition hang or a system outage due to electrical failure. The probability that such an event will cause a problem is primarily dependant on system configuration as discussed in “Large Number/Complexity of Files Defined in a Catalog” on page 200

The most common causes for catalog corruption are improper migration from one device type to another and improper data sharing configuration. VSAM migration will not be discussed here (it would require a chapter by itself) and proper data sharing configuration will only be briefly discussed. Other potential sources of VSAM catalog corruption are discussed, however, it is usually very difficult to identify the particular cause of a catalog corruption.

Unfortunately, catalog corruption is not necessarily immediately obvious. The catalog is in most cases still usable except for a few particular files. However, catalog corruption is self-propagating. Once a catalog is corrupted it only gets worse. Once a catalog corruption is detected, the catalog should be rebuilt as early as possible to prevent the damage from spreading to unaffected areas of the

catalog. In any case all precautions that are discussed in this article should be considered for implementation.

Cross-Systems Sharing

VSE/VSAM allows for the sharing of catalogs and files across z/VSE systems. For sharing among systems, you must establish the DASD sharing environment through the correct system generation and IPL commands. The catalogs and files must reside on SHARED DEVICES.

You do not specifically invoke cross-system sharing when opening catalogs and files, because.

- Catalogs are automatically shared if they reside on shared devices.
- Files are automatically shared if they are owned by a shared catalog.

Basically to ensure correct sharing of VSAM catalogs.

- All VSE systems must share the same lock file
- All shared DASDs must be ADDed with the SHR option on all systems having access to the volume
- Under VM.
  - all shared MDISKs must be defined with MWV on the MDISK statement
  - all VSE systems must LINK..MW to that MDISK
- There can be NO DUPLICATE VOLIDS on the system because VSAM generates lock resource names using the catalog's VOLID.
- In a DASD sharing complex all VSE systems must be properly shut down to release all locks (End-of-task in all partitions). Otherwise the UNLOCK AR command may be used to free outstanding locks, or all systems must be shut down, and the first system to re-IPL should have a DLF...TYPE=F statement.

Large Number/Complexity of Files Defined in a Catalog

A large number of files defined in a catalog is not directly a cause for catalog corruption, but can make recovery from a corrupted catalog more difficult and can possibly increase the probability of a catalog corruption. The number of files defined in a catalog have a direct impact on the performance of VSE/VSAM activities.

A large number of files in a single catalog (for example, a thousand files) can significantly increase the run time for most IDCAMS functions. This includes DEFINE, DELETE and LISTCAT functions. It also impacts open and close performance. The longer the run time the greater the window of opportunity for an I/O error.

The exact number of files at which the impact on performance becomes noticeable depends on several factors (for example, DASD access speed and file name pattern). As the number of files in a single catalog increases, you should carefully monitor the performance of the indicated IDCAMS functions. The main factor, however, is the complexity of the files in the catalog. What is meant here is the length of the associations for the file in the catalog. Satisfactory performance may be experienced with 500 or more files in a catalog when they all only have space on a single volume. On the other hand fewer files that have space on multiple volumes (even as CANDIDATE volumes, possibly as a result of a DEFAULT

MODEL) could result in terrible performance. Multiple extents for each file make the situation even worse.

Each operation on one of these “complex” files requires significantly more operations and much more time. This provides an even larger window of opportunity in which an operation can fail.

Rogue Programs

There is also the possibility that a user application can cause catalog corruption. One possibility is when an application generates its own channel programs to write to DASD. This could result in the application inadvertently writing directly over catalog information on the DASD itself. It is also possible that an application can mistakenly access memory that does not belong to it. In this case, it would be conceivable that VSAM control blocks could be overwritten, and catalog information could be subsequently updated with incorrect values.

Catalog Precautions

Using Read-Only Catalogs in a z/VSE Environment

A minidisk under VM can be linked in read-only mode, that is, the virtual machine is not allowed to write to this minidisk. VSE/VSAM support of catalogs that reside on read-only disks provide a limited way to share data between a z/VSE guest machine and a CMS or GCS guest machine running under the same CP complex. With this support.

- Only one virtual machine has write access to the catalog disk, and any other virtual machine has read-only access.
- If the data set is cataloged in a read-only catalog, only OPEN for input (MACRF=IN the ACB) is permitted.
- Statistic (“REC-RETRIEVED”) will not show how many records are retrieved, because catalog information cannot be updated in a read-only catalog.

Normally, z/VSE determines whether a minidisk is linked in read-only mode during IPL and DVCUP processing. To change the access of a z/VSE system to a catalog volume from read/write to read-only mode proceed as follows.

1. From the z/VSE system, issue the DVCDN job control command against the device, followed by the OFFLINE attention routine command, for example.

   ```
   DVCDN 400
   OFFLINE 400
   ```

2. From the virtual machine console of the z/VSE guest machine, link the minidisk with the correct mode (usually RR for read-only and MR for write access), for example.

   ```
   #CP LINK VSESYST 400 400 RR
   ```

3. From the z/VSE system, issue the ONLINE attention routine command, followed by the DVCUP job control command against the device., for example.

   ```
   ONLINE 400
   DVCUP 400
   ```

To check which volumes are currently treated as read-only devices, the z/VSE operator can issue the VOLUME attention routine command.

Notes:

1. Once VSE/VSAM has opened a catalog, it remains open until end of job step. Consequently, a volume on which a catalog resides may not be changed to a read-only status while the catalog is opened from any partition. If a catalog volume is changed from read-only to read/write mode while the catalog is opened from a partition, no output OPENs are permitted from this partition until the job step ends and the catalog is re-opened.
2. Only the IDCAMS functions LISTCAT, PRINT, REPRO and EXPORT TEMPORARY can be used with R/O catalogs.
3. If a cluster is exported through EXPORT TEMPORARY from a R/O catalog, then the cluster entry in the catalog cannot be marked as being temporary exported. The cluster must be deleted before the copy can be imported again.

Catalog Organization

Note - for additional information on "Catalog Organization" see p.27 of the presentation "Design & Tuning of VSE/VSAM" from the *"2002 z/VM, VSE and Linux on IBM zSeries Technical Conference", Oct. 2002* , available at the location: ftp://ftp.software.ibm.com/eserver/zseries/zos/vse/pdf3/techconf2002/E54.PDF Note - for recommendations on "CI vs CA splits" see pp.28-30 of the above reference.

Generally, it is recommended that all the VSAM space on a particular volume belong to one catalog. If practical, it is also recommended that every volume have its own catalog (as described in “VSE/Fast Copy” on page 203). This negates VSAM's ability to manage multiple volumes through a single catalog, however, it simplifies recovery drastically.

If a system contains only one (master) catalog and that catalog is destroyed, the resources of the whole system are lost and must be restored by the use of backup copies. When several user catalogs are used, only the resources controlled by the destroyed catalog are affected, and it can be rebuilt while processing on other data continues.

The DEFINE USERCATALOG command can be used to create many user catalogs (as many as one per volume) and reduce the number of files per catalog. If a catalog becomes unusable and has, for example, only ten files *cataloged in it, access to only those ten files has to be recovered.

If the master catalog is destroyed, you need only rebuild the master catalog and the resources directly connected to it because user catalogs (like the master catalog) are self-describing. You only need to rebuild the master catalog and re-establish access to the user catalogs through IMPORT CONNECT.

A bottom line recommendation, considering the aspects mentioned in “Large Number/Complexity of Files Defined in a Catalog” on page 200, is 300-400 files per catalog. This should provide acceptable performance in most cases.

Catalog Backups

Because VSE/VSAM catalogs are so important, you should consider them carefully in your backup strategy. If all of the files owned by a catalog are backed up individually, it is possible to recover from a destroyed catalog by restoring all files to a new catalog. The probability of losing an entire catalog is very low. However, to speed recovery or minimize exposure in the case of catalog damage or destruction, several backup methods are available.

VSE/Fast Copy: You can use VSE/Fast Copy to create a backup copy of an entire volume and restore that copy back to a volume. However, all volumes controlled by a VSE/VSAM catalog must be backed up and restored together. Otherwise, inconsistencies between the catalog and the physical data volumes may result in data corruption.

IDCAMS Backup / Restore: This is the preferred method to back up VSE/VSAM clusters. Individual clusters can be restored from a back-up tape, as required. Multiple catalogs maybe backed up onto a single tape. This does, however, require special handling to recover files (see separate section).

REPRO: can be used to reorganize individual files. The advantage over IDCAMS Backup / Restore is that the unload and load is performed at the record level, which reorganizes the file (serializes records and places them in contiguous extents). The disadvantage is that it is substantially slower, and requires that the file definition be performed separate from the load process.

REPRO the catalog itself does not backup all the files in a catalog; only catalog data is backed up (cluster definitions), not the data. However, under certain situations (see Section *Catalog / File Prematurely Full*), REPRO can be used to reorganize a catalog. The following restrictions apply:

- The catalog should be unloaded (REPRO'd) to a sequential file (such as tape), then immediately re-loaded. No files in the catalog should be updated between the unload and reload steps.
- Do not delete the catalog between the unload and reload step. Simply reload the catalog right over top of the previous copy. There is special code in IDCAMS REPRO which detects this condition, and performs a reorganization of the catalog.
- The catalog may be loaded over a newly-defined catalog of a larger extent. This is a quick way of enlarging a catalog, but all clusters defined in the previous catalog on the catalog volume will be flagged as “not useable” and must be restored.

Catalog Check

The CATALOG CHECK SERVICE AID (IKQVCHK) helps you to determine whether a catalog has been damaged and, if damaged, the type and extent of the damage. You should always run IKQVCHK to assess catalog integrity in the following circumstances.

- After a system failure.
- When a file or catalog does not behave as expected.
- As part of regular system maintenance.

It has also been recommended to incorporate IKQVCHK into normal operating procedures in some manner. This could take the form of running IKQVCHK once every morning and once every evening to validate the catalogs integrity prior to daily interactive processing and nightly batch processing.

IKQVCHK can not, however, detect space allocation mismatches in a catalog. If this is believed to be the case, there are diagnostic programs available from VSE Level 2 support that can perform detailed analysis of LISTCAT output. This is only recommended in cases where the cause of repeated catalog corruption can not be satisfactorily identified.

DASD File Protection

The DASDFP operand of the IPL SYS command can be used to specify whether disk file protection should be active. If you omit the operand, the system does not activate file protection. Using DASD file protection may only be beneficial for applications that generate their own channel programs as discussed above in “Rogue Programs” on page 201.

The DASD File Protection facility prevents programs from writing data outside the limits of their disk files. This might happen if, for example, a randomizing algorithm produces an unexpected disk address which is outside the file limits. Other files on a disk volume are thus protected against unintended destruction. However, if two disk files have been opened in the same partition and use the same programmer logical unit, these two files are not protected against destroying each other's data.

Miscellaneous Problems

Read Integrity Problems

Note - for information on

- "illogic error" symptoms (for example, concerning alternate indices and base cluster updates), see p.60
- further integrity problems than discussed here, see pp.60-62

of the presentation "Design & Tuning of VSE/VSAM" from the *"2002 z/VM, VSE and Linux on IBM zSeries Technical Conference", Oct. 2002* , available at the location:
ftp://ftp.software.ibm.com/eserver/zseries/zos/vse/pdf3/techconf2002/E54.PDF

Basically, read integrity problems arise whenever you have one program updating a file, while simultaneously, another program is reading the same area of the file. It doesn't make any difference whether these are two programs (transactions) in the same CICS partition, two programs in different partitions on the same VSE machine, two different VSE systems under VM, or two different physical CPUs. The problem is that a read task cannot see the updated control interval, until it is physically written to disk. Unfortunately, with SHR(2), various file components (data, index, alternate index) as they are updated, are only written to disk when VSAM needs the buffers for another I/O, so are not necessarily always “in synch”. This can cause a number of problems to a read task, including missing records, I/O errors, loops, or return code x‘90’ (indicating an alternative index pointer without a corresponding base record). There are several conditions which impact the frequency of read integrity problems.

1. Using SHR(4) results in records being written to disk much more frequently than SHR(2), whose updates may not be written to disk until close time. Normally, converting a file to SHR(4) will resolve most read integrity problems. However, this is done at the cost of much more I/O, since most record reads and updates are now read/written directly to/from disk, ignoring the content of the current buffer.
2. VSAM only guarantees read integrity if the user performs a “Get for Update”. This will ensure that the record is read from disk, not from a buffer already in the user storage. The only other condition where read integrity is guaranteed, is if the file is defined as SHR(1). This means if someone has the file open for output, no one else can open the file, even for input.

3. Records written to a file during initial file load (until first OPEN), are not available until the file is closed for the first time. Regardless of the defined file shareoption, VSAM opens an empty file for output as SHR(1). This prevents all other OPENs until at least one record has been written to the file, and it is closed.
4. Records updated using “Sequential Get for Update” may not be written out immediately upon being changed, even with Shareoption(4) files.
5. If a file frequently suffers read integrity problems, all update tasks should issue periodic ENDREQ commands. This “flushes” the buffers, and ensures a consistent file condition. It is used by an update transaction to complete processing on a string, and ensure that all changed buffers are written to disk. You do not have to close the file. Under CICS, see the UNLOCK command. See also *CICS/TS Problem Determination Guide.*
6. If the read and write task are both within the same partition, some alleviation of the problem is possible without using “Get-for-Update”. Opening the file as “Dataset Name Sharing” shares the buffers between all ACBs, thus giving the read task direct access to the CI in the write task's buffer chain, before it is written to the disk. (This is particularly helpful for files which are accessed / updated over both the base and an alternate index). Defining the file as LSR, has much the same effect. This is not a complete fix, however, it only shortens the window.
7. Defining MACRF=(...DFR...) (Deferred Write) will result in records not being written out until a WRTBFR command is issued.

In addition, the problem is complicated by the rules for “look-aside”. “Look- aside” is a VSAM function, where, before reading a record in from disk, VSAM buffer management first checks to see if that record is already in storage. The following rules apply for “look-aside”.

1. VSAM can only do “look-aside” in the buffers attached to this ACB. Thus, when you use LSR, this increases the pool of buffers available to VSAM to search. Remember, LSR is only shared within a partition, not within a machine, or between machines. However, retrieval of a control interval from an LSR pool (even in read mode), will lock all other CICS transactions from retrieving this same control interval. This is not true of private buffers.
2. With Shareoption(1), since no one else can have the file open for write, while I have it open for either read or write, VSAM always does “look-aside” (with the restrictions indicated in point 1).
3. When records are read in sequential mode, VSAM maintains its positioning in the current control interval, and always checks there for the next record, before initiating a new I/O to disk.
4. With direct retrievals in a Shareoption(2) file, VSAM only does “look-aside” if I am the one with the file open for output. Otherwise, doing “look-aside” if I only have the file open for read means that I might miss an update submitted by another program with the file open for write.
5. With direct retrievals in Shareoption(3) or (4) files, VSAM does not do any “look-aside”, since the record may have already been updated by another program in a set of buffers to which I do not have access.

The emphasis in VSAM is on file integrity, not necessarily access speed.

Catalog / File Prematurely Full

Any KSDS file (including a VSAM catalog) can become prematurely full due to "wasteland" in the index. This is particularly true if the file has experienced a large number of adds and deletes, where whole ranges of records (or files, in the case of a catalog) have been deleted. Once VSAM fills up a control area and extends the file into a new control area, if all records in the previous control area are deleted, VSAM will not reclaim that space. If a new record (or file, in the case of a catalog) is added, containing a key in the range previously serviced by that control area, VSAM will place the new record / file in the other-wise empty control area. Otherwise, it remains forever "wasteland". The impact on performance comes when we do sequential retrievals or searches. Since there is no indication in the index that the control area is empty, VSAM must read each data control interval and check it, only to discover that there are no data records there matching the criteria. This can lead to excessive I/O and performance degradation.

To alleviate this situation, the file or catalog must be unloaded and reloaded using IDCAMS REPRO. See previous section under "Catalog Backups" on page 202.

MSG 4226I / 4227I ("Automatic Close")

Normal Automatic Close messages are:

```
msg4A87I   Automatic Close has been started
msg4A88I   Automatic Close for nn files completed
```

These are warning messages that a file (ACB) was not closed prior to the program going to end-of-job. This is normally not a problem, unless msg4A87I is followed by one of the following:

```
Msg 4226I   Automatic Close could not be started. File = <..> R=<rc>
Msg 4227I   Automatic Close was not successful. File = <..> Close err code =
```

These last two messages should always be taken seriously, as they normally indicate overlay of VSAM control blocks. The file remains in an uncertain condition, with the statistics not having been updated, and, potentially, file updates missing. To debug this condition, however, an AR DUMP of the partition is require PRIOR to the program going to end-of-job. A dump after the message will only show Job Control in the partition.

If you are experiencing these errors, a patch is available from VSE/VSAM support to assist you in identifying which files are not being closed.

Default Models

Default Models can be used to catalogue a pattern which can be mapped to future cluster definitions. The most common application is for implicitly defining SAM ESDS files:

```
// DLBL IJSYS01,'DOS.WORKFILE.SYS001',,VSAM,RECORD=(1000,500),RECSIZE=(512)
```

The size of the file to be defined is calculated by multiplying the recordsize by the number of records, but VSAM needs to know where the space should be allocated. This is passed to VSAM via the default model:

```
DEFAULT.MODEL.ESDS.SAM.
```

Missing Default Model

The following series of messages appears rather intimidating, but in actuality only mean that you tried to open a SAM ESDS file (new file for output) without a DEFAULT.MODEL defined in the catalog (or perhaps included a VOLID on the // EXTENT card which did not match the candidate volumes in the default model.

```
4A37I FILE SAMFILE  CATALOG ERROR DURING IMPLICIT DEFINE - 248 , BX ,000
4228I FILE SAMFILE    OPEN  ERROR X'4F' (079) CAT=UCAT    (  8 , CG ,  6 )
(IKQOPN-3) RC X'0000004F' ON CALL TO  IKQOPNHC
0V15I REQUEST FROM SYSTEM SERVICE ROUTINE
0S08I LOG. TRANS. AREA CANCELED, PHASE = $$BOSMXT
0S00I JOB RUN      CANCELED.
1I51I DUMP COMPLETE
1S78I  JOB TERMINATED DUE TO  PROGRAM ABEND
EOJ RUN
```

Figure 88. Messages Received for Missing DEFAULT.MODEL

Where:

- **248** indicates "Volume Record not found"
- **BX** is IGG0CLBX, the VSAM Catalog Mgmt routine that caught the error
- **X'4F'** indicates an error during implicit define
- **8** and **6** indicates a no-record-found condition, in this case, the search for the default model
- **CG** is, of course, IGG0CLCG
- **$$BOSMXT** is the SAM interface to VSAM, so it is the one that catches any errors

Using a Reserved Name

The following message indicates that you attempted to define a cluster using IDCAMS with a reserved (default model) name. Default models must be defined as NOALLOC, REUSE files.

```
IDC3299I INVALID PARAMETER COMBINATION: RESERVED NAME/SUBALLOC/UNIQUE
IDC3003I FUNCTION TERMINATED. CONDITION CODE IS 12
```

Defining a Default Model

The following should be used to define a default model for SAM ESDS:

```
// EXEC IDCAMS,SIZE=AUTO
    DEFINE CLUSTER                          -
        (NAME(DEFAULT.MODEL.ESDS.SAM)       -
        NOALLOCATION NONINDEXED REUSE       -
        SHAREOPTIONS(1,3)                   -
        VOLUMES(SYSWK1,DOSRES)              -
        TRK (1 1) RECORDFORMAT(U) UNORDERED -
        RECSZ(2000 2000))                   -
      CATALOG(...)
    LISTCAT ENTRIES (DEFAULT.MODEL.ESDS.SAM) ALL CATALOG(...)
/*
```

Figure 89. Define a DEFAULT.MODEL (SAM ESDS) Sample Job

Recovering From a Physical Medium Failure

1. Keep your user catalogs small.

   One user catalog per volume, if possible. Use Fast Copy (or a similar full pack backup utility) to copy the entire catalog volume (including catalog and all files) to tape.

   Recovery: Requires a simple copy of the entire volume back to the replaced drive. The target volume must exactly match the characteristics of the original drive. See “Return Code 254 x‘FE’” on page 193.

2. If all files in the catalog are critical and require daily backup:

   - Use IDCAMS BACKUP (or a similar utility) to individually backup all files in the catalog.

   Recovery:

   - Perform an IDCAMS EXPORT <user catalog> DISCONNECT, to remove the user catalog pointer from the master catalog.
   - Initialize all volumes belonging to the user catalog by deleting all VSAM files on those volumes and resetting the “volume ownership” flag. This may be done by re-initializing the pack or using a utility such as DITTO or IKQVDU.
   - Define a new user catalog and associated space.
   - Restore all files from backup tapes.

3. If only a small number of the files in the catalog require daily backup, and the catalog controls a large number of volumes:

   - Keep full-pack backups of all volumes of the catalog
   - Use IDCAMS BACKUP (or a similar file backup utility) to backup critical files daily.

   Recovery:

   - Restore the full pack copy of the volume which was replaced.
   - Restore all critical files with at least one extent on the volume which was replaced.

4. If you must replace a volume other than the one containing the user catalog, and do not wish to re-build the entire catalog: This requires removal of the old volume entry from the user catalog, and definition of a new one. The problem is that IDCAMS will not allow a DELETE SPACE unless the volume, containing the VSAM extents, is physically on-line. We therefore have to trick IDCAMS into thinking the old volume is still accessible.

   Note: For this condition, do not attempt DELETE SPACE FORCE or DELETE CLUSTER IGNOREERROR as this will only dig you deeper into the hole.

   - Define a new user catalog on a different volume, which does not already contain a user catalog. Call the previous user catalog UCAT1. Call the new (temporary) user catalog UCAT2.
   - Using IDCAMS SPACE, define a space in UCAT2 on the replaced volume, which matches, exactly, the original space still defined in UCAT1.
   - From UCAT1, delete all clusters on the replaced volume, and the space. If UCAT2 is corrupted, and you are unable to delete the space, you have no

choice at this point but to re-build the entire user catalog. For this reason, we recommend that you run IKQVCHK at least monthly against all VSE/VSAM catalogs.

- Remove the pointer to UCAT2 from the master catalog using EXPORT <ucat2> DISCONNECT
- Physically remove the VTOC extent for UCAT2 using DITTO or IKQVDU.
- Re-define the space on the replaced volume in UCAT1, and restore the affected files.

CA Splits

Insertion of a record into a file may result in a Control Interval (CI) or Control Area (CA) split. During a CA split, CIs found after the insertion point in the original CA are moved to a new CA physically located at the end of the last extent of the file. There is a window of time where the same CI actually exists in two CAs at the same time. To keep track of this condition, VSAM first marks all CIs in the old CA to indicate they are being moved. This is done via a “CA-split-in-progress“ bit located in the CIDF (last 4 bytes in the CI). If, prior to completing the move, the CA split is interrupted for any reason (program check, operator-initiated cancellation, I/O error, etc), the file is left in an incomplete condition. A cancellation during this window may cause duplicate record conditions, VSAM-reported I/O errors, or loops during sequential access. Many backup/restore products check for the CA-split-in-progress bit and log an error during backup.

The problem may be corrected as follows:

1. CA-splits may be minimized by defining FREESPACE during CLUSTER definition.

   Note: Free Space is only used during initial load mode, or new extent allocation.

2. If the error is being detected by a backup utility, ensure the file being backed-up is closed in all other partitions. Otherwise, the possibility exists that you have detected a CA-split in progress. You can also test this by re-running the backup job after waiting a suitable interval for the split to finish.

3. VSAM has automatic recovery code built in to correct this condition under certain conditions. Basically, the file must be open for output, and VSAM must perform a get-for-update in sequential mode of one of the CIs with the CA-split-in-progress bit turned on. When VSAM detects a data control interval with the “ci-split-in-progress” bit on, it calls IKQDDR, under the covers, which reads the previous CA and new CA, and attempts to complete the move. It is often possible to trigger this correction by writing a simple utility program which reads the entire file in sequential mode for update. After running this utility, re-run REPRO to verify that the condition has been corrected.

4. Failing the above, the file must be re-built. Many customers have had success by accessing the data component of the file (as an ESDS file) instead of the cluster. The data component is copied to a sequential dataset (using REPRO), sorted according to the primary key, duplicate records eliminated, then re-loaded at the cluster level.

5. Or you can always go back to the last valid backup.

   If additional PD/PSI is desired, use DITTO DD, to print off the record with the CA-split-in-progress bit. The last 7 bytes of the CI should contain a valid RDF

and CIDF. *See "VSE/VSAM Logic" (LY33-9130)*, page 5-2). X"80" in the next-to last byte of the control interval indicates "CI-split-in-progress".

Missing or Unreadable Print Information

Since VSE/VSAM printing of files is done in upper case characters without respect to, for example, lower case or foreign language special characters, some information may not be readable. This problem may be solved by a user-provided translate table for IDCAMS PRINT. The IDCAMS PRINT command is a basic method of listing records from a VSAM file. Through the use of the **PARM** parameter processing options for printed output can be specified.

The **GRAPHICS TABLE(mname)** parameter of the **PARM** command specifies the name of a module (accessible through VSE/VSAM CDLOAD) in the sublibrary that contains a 256-byte user-provided translate table. This table defines the graphic characters for each of the possible 256 bit patterns. Any character to be printed is translated to the bit pattern found in such a table at the position corresponding to its numeric value (0-255).

Restriction: If you specify the TABLE parameter, issue the PARM command early in the IDCAMS command stream to avoid having CDLOAD fail. An example of such a module that allows printing in upper and lower case is as follows:

```
* $$ JOB JNM=VSAMTRT,DISP=D,CLASS=0
// JOB VSAMTRT
// OPTION CATAL
// LIBDEF PHASE,CATALOG=IJSYSRS.SYSLIB
   PHASE VSAMTRT,*
// EXEC ASSEMBLY
VSAMTRT  CSECT
         DC    CL16'                '   hex 0X
         DC    CL16'                '   hex 1X
         DC    CL16'                '   hex 2X
         DC    CL16'                '   hex 3X
         DC    CL16'           .<(+|'   hex 4X
         DC    CL16'&&          $*); '   hex 5X
         DC    CL16'-/         ,%_>?'   hex 6X
         DC    CL16'          :#@''="'   hex 7X
         DC    CL16' abcdefghi      '   hex 8X
         DC    CL16' jklmnopqr      '   hex 9X
         DC    CL16'  stuvwxyz      '   hex AX
         DC    CL16'                '   hex BX
         DC    CL16' ABCDEFGHI      '   hex CX
         DC    CL16' JKLMNOPQR      '   hex DX
         DC    CL16'  STUVWXYZ      '   hex EX
         DC    CL16'0123456789      '   hex FX
         END
/*
// EXEC LNKEDT
/&
* $$ EOJ
```

Figure 90. Sample to print in upper and lower case

An example of such a module that allows printing in upper and lower case with German characters enabled is as follows:

```
* $$ JOB JNM=VSAMTRT,DISP=D,CLASS=0
// JOB VSAMTRT
// OPTION CATAL
// LIBDEF PHASE,CATALOG=IJSYSRS.SYSLIB
   PHASE VSAMTRT,*
// EXEC ASSEMBLY
VSAMTRT  CSECT
         DC    CL16'                '   hex 0X
         DC    CL16'                '   hex 1X
         DC    CL16'                '   hex 2X
         DC    CL16'                '   hex 3X
         DC    CL16'          Ä.<(+|'   hex 4X
         DC    CL16'&&         Ü$*); '   hex 5X
         DC    CL16'-/        ö,%_>?'   hex 6X
         DC    CL16'          :#@''="'   hex 7X
         DC    CL16' abcdefghi      '   hex 8X
         DC    CL16' jklmnopqr      '   hex 9X
         DC    CL16' ßstuvwxyz      '   hex AX
         DC    CL16'                '   hex BX
         DC    CL16'äABCDEFGHI      '   hex CX
         DC    CL16'üJKLMNOPQR      '   hex DX
         DC    CL16'Ö STUVWXYZ      '   hex EX
         DC    CL16'0123456789      '   hex FX
         END
/*
// EXEC LNKEDT
/&
* $$ EOJ
```

Figure 91. Sample to print in upper and lower case German

The following JCL demonstrates how to use the **PRINT** command with this **PARM** parameter:

```
* $$ JOB JNM=FILEPRT,DISP=D,CLASS=0
// JOB FILEPRT
// DLBL UCAT,'EXAMPLE.USER.CATALOG',,VSAM
// DLBL KSDS,'SAMPLE.FILE',,VSAM,CAT=UCAT
// EXEC IDCAMS,SIZE=AUTO,PARM='GRAPHICS(TABLE(VSAMTRT))'
   PRINT INFILE (KSDS) DUMP
/*
/&
* $$ EOJ
```

Figure 92. Sample how to use the PRINT command

VSE/VSAM Enhancements (z/VSE Version 5)

The following section describes enhancements to the VSE/VSAM component along with documentation references where a more detailed explanation is available. These enhancements came with z/VSE Version 5.

SHOWCB Enhancements

Starting with z/VSE 5.1 VSAM provides a list of new SHOWCB fields. Among that LSR Matrix and Extent Matrix can be requested for detailed information about LSR pool or VSAM dataset extents. The SHOWCB macro now supports the following set of new fields/attributes for the ACB :

BFREE	Number of unassigned buffers.
CDBUF	Number of data buffers.
CIBUF	Number of index buffers.
CIPCA	Number of control intervals per control area.
CNAME	Name of the cluster (44 bytes).
EXTINF	Refer to Extent Matrix description.
IDACB	The ACB identifier is equal to x'A0'.
IDDOS	The DOS identifier is equal to x'28'.
LNEST	Local number of index levels.
OPENOBJ	AMS flag byte. With the AMS flag you can determine whether the opened object is a path, a base cluster, or an alternate index: • x'80' = alternate index • x'40' = access via path • x'20' = access via base cluster

For description of LSR Matrix or Extent Matrix as well as for other SHOWCB enhancements refer to *Chapter 12 "Descriptions of VSE/VSAM Macros"* and *Appendix A "Operand Notation and Parameter Lists for VSE/VSAM Macros"* in the *"VSE/VSAM User's Guide and Application Programming".*

IDCAMS Commands Security

z/VSE 5.2 provides the ability to control user access rights to the IDCAMS Commands. Thus System administrators can restrict the usage of IDCAMS commands with the help of a security manager, for example the Basic Security Manager (BSM) provided with z/VSE. See the new section *IDCAMS Commands Security* of *Chapter 8 Data Protection and Data Recovery* in the *"VSE/VSAM User's Guide and Application Programming"* manual.

DLBL CICIZE parameter for SAM-ESDS Implicit Definition

Starting with z/VSE 5.2 the DLBL CISIZE parameter is no longer restricted to SD files but can also be used for VSAM files. This allows the customer to specify a CISIZE other then the default assigned by VSAM for SAM-ESDS files implicitly defined via DLBL.

See CISIZE operand description in *Format of the DLBL Statement'* section of *'Chapter 3 Operation and Job Control'* in the *"VSE/VSAM User's Guide and Application Programming"* manual.

Chapter 9. VSE/ICCF

Submit to Batch

To transfer z/VSE job streams from the ICCF library (DTSFILE) to VSE/POWER, ICCF provides SUBMIT procedure and program DTSSUBMT. This program is invoked by entering 'SUBMIT member' on the ICCF command line.

If you submit to batch from IUI panel IESLIBP, IUI's submit program (DTSIESSS) is used. There are substantial differences between the two submit programs. ICCF performs an extensive validation of POWER JECL operands depending on the user's authorization. (OPT A Bit 7 in the user's profile). The ICCF SUBMIT procedure can be modified to direct the job's to certain classes, assign specific dispositions or generally control any POWER JECL operand. DTSIESSS leaves the validation more or less to POWER.

If bit 7 of the OPTA option byte = on, user's POWER JECL specifications are not overwritten by the SUBMIT procedure values. OPTA bit 7 can be specified via DTSUTIL ALTER USER command.

If bit 7 of the OPTA byte = off, user's POWER JECL specifications will be overwritten by the SUBMIT procedure values. This is true even for a specified jobname which will be replaced by the ICCF member name. To allow the specification of certain operands by a 'OPTA_bit_7_off' user, this operand's value in the SUBMIT's procedure's model statement must be specified in parentheses.

Model: * $$ PUN DISP=(D),CLASS=(A)

User: * $$ PUN DISP=H

To Power: * $$ PUN DISP=H,CLASS=A

To see the statements which are really transferred to POWER use the PRINT option of the SUBMIT procedure:

SUBMIT member PRINT

ICCF Library

Increase Number of Users and Libraries

The easiest way to increase maximum number of users and libraries is during DTSUTIL RESTORE. FORMAT is not necessary if only the numbers should be increased and the extent location and size remain the same.

RESTORE Libraries(200) Users(200)

Each new library requires a library header record to be established

ADD Library Maxdir(n) Freespace(percent) Date

Since VSE/ESA 2.x there is a DTSUTIL command available which returns maximum number of users and libraries specified.

DISPLAY Format

Increase Maximum Number of Members per Library

DTSUTIL ALTER LIB(nn) MAXDIR(mm)

This command can be issued while the DTSFILE is connected.

DTSFILE Extension

Indications for a necessary DTSFILE extension are messages

- K088I HI FILE RECORDS = (0) or close to 0 especially after restoring the DTSFILE
- Message * LIBRARY FILE IS FULL

Follow carefully the instructions given in the "z/VSE Administration" manual when extending the DTSFILE.

UPIP Flag

The Update_in_progress flag is set for the member being edited. If another user tries to edit the same member message '*UPDATE IN PROGRESS-TRY LATER' is displayed. There are cases where this flag isn't reset like system crash or after a PEND IMM. The administrator has several ways of clearing this flag.

```
/PROTECT member UPIP
DTSANALS RESET UPD
```

If the DTSFILE has not been closed correctly, normally during the next startup of CICSICCF 'DTSANALS RECOVER OPT' will cause the recovery job to run and reset all UPIP flags. There are some cases where RECOVER OPT finds the DTSFILE doesn't need recovery, but there are still members with UPIP on. If you are facing such events you could use 'DTSANALS RECOVER OPX'. This causes DTSANALS to scan all member directory records for UPIP flags, regardless if the file is intact. The flags are reset and a message for which member the flag has been cleared is written. Otherwise OPX is exactly the same as OPT. This function has been introduced with APAR PN80038 for VSE/ESA 2.1/2.2.

Compressed Members

Library members which have been compressed via the /SQUEEZE command can be converted back by entering input mode and using the /INSERT command. Here is a procedure for single members and a macro to uncompress multiple members of a library.

```
Syntax: PUNC membername passwrd
 * **************
 * PROCEDURE TO UNCOMPRESS
 * **************
 &&OPTIONS 0010011
 /INPUT
 /INSERT &&PARAM1 &&PARAM2
 /END
 /REPLACE &&PARAM1 &&PARAM2
 &&IF &&RETCOD NE *REPLACE &&GOTO FAIL
 &&TYPE MEMBER &&PARAM1 IS NOW UNCOMPRESSED
 &&EXIT
 &&LABEL FAIL
 &&TYPE ERROR OCCURRED ... MAYBE MEMBER &&PARAM1 NOT FOUND
 /PEND
 /RUN
```

Figure 93. ICCF procedure PUNC to uncompress a member

```
@MACRO
@NOPRINT
PUNC CMEMB1
PUNC CMEMB2
PUNC CMEMB3
```

Figure 94. ICCF macro MUNC for multiple uncompress members

To get all compressed members of a library into macro MUNC you could:

1. Run ICCF procedure SDSERV. This will create member $$PRINT (ICCF's print area)
2. Enter ED and GET $$PRINT
3. Delete messages
4. FILE membername
5. Run 'SORT membername SEQ D3703 PUNCH membername2' if 2 digit year
6. Run 'SORT membername SEQ D3903 PUNCH membername2' if 4 digit year. Will sort for attribute CPR.
7. Edit member2 for macro MUNC
8. Run MUNC

GETVIS Subpools

VSE/ICCF uses GETVIS SUBPOOL IDs to better identify VSE/ICCF as storage owner and to better analyze GETVIS below shortages in the CICS/ICCF partition. The IDs used are as follows:

1. ICCFFN VSE/ICCF needed for ICCF shutdown - no reduction possible.
2. ICCFCN VSE/ICCF control program - no reduction possible.
3. ICCFIP VSE/ICCF interactive partitions - reducing its numbers or size would decrease GETVIS requirements.

4. ICCFBU VSE/ICCF buffers - reducing its numbers or size would decrease GETVIS requirements.

View Library Members

A programmer user *test* should be able to have read access to library members from other users.
Library 100 is used.

- library 100 is defined as PUBLIC, accessible for all
- the user *test* creates all members with the attribute PRIVATE.
  This is done automatically.
  If not, define the user with OPTA bit 5=1

bit 5 = 0 When this switch is off, the user's library members will be saved as public unless specified as private in the /SAVE command. dt.bit 5 = 1

The user's library members will be saved as private, except if specified otherwise in the /SAVE command.

All other users can access library 100, but members which are marked private and owned by TEST can be read only.

Chapter 10. REXX/VSE

Setup

Initialization

Most of the REXX/VSE programs run only successfully, if the REXX/VSE initialization step has been executed in advance. The REXX/VSE initialization step is invoked via "// EXEC ARXLINK" and is usually contained in system startup procedure USERBG.PROC .

If problems occur with REXX commands or functions requiring one of the external or replaceable REXX routines, run the REXX/VSE initialization step first and try to rerun the REXX program.

ARXLINK RC=21

If the REXX initialization step ARXLINK returns with a RC=21, this means that a problem has occurred when trying to do a CDLOAD for one of the phases loaded during REXX initialization.

A CDLOAD may fail, if

- there is not sufficient GETVIS storage in the partition,
- but also if the phase to be loaded is not found within the active phase chain.

Phases to be loaded are for instance: ARXIOLAR, ARXPARMS, ARXANCHR, ARXOUT, the host command environment phases mentioned in ARXPARMS, and the function package phases mentioned in ARXPARMS.

GETVIS Size

You cannot run your REXX/VSE programs in partitions offering only a GETVIS size of 256K. In this case processing stops immediately with the following messages:

```
ARX0565I LIBRM ERROR OPEN RC=(0020,0100) ARXIOLAR
ARX0110I The REXX exec cannot be interpreted.
ARX0112I The REXX exec cannot be loaded.
R003I REXX/VSE EXEC PROCESSING FAILED, RETURN CODE 20
```

If you use a partition with a higher GETVIS size, the REXX interpreter starts at least processing. The finally required GETVIS size depends on your REXX program (for instance on the space required for stack contents and REXX variables).

Setup for Using the REXX/VSE SOCKET Function

There exist 2 implementations of the REXX SOCKET function with different functionality: One is from IBM and provides the standard low-level Socket interface available on many platforms, the other is from CSI with a CSI-specific interface on a higher level with more functionality.

The CSI SOCKET function is implemented within SOCKET.PHASE and can be used without additional configuration steps. The IBM SOCKET function is

implemented within function package ARXEFSO, which needs to be separately configured using REXX/VSE customization job ARXPARMS.Z (see SOCKET-specific comments). The single steps are described in detail at the end of Chapter 15 "REXX Sockets Application Program Interface" in the REXX/VSE Reference Manual.

To allow coexistence of both SOCKET functions, it is possible to rename the IBM SOCKET function to "SOCKEN" and activate function package ARXEFSN instead of ARXEFSO.

To use the IBM SOCKET function, you have to invoke subfunction 'INITIALIZE' first. If this invocation returns only an empty string, the setup for the IBM SOCKET function is not correct, because the CSI-delivered SOCKET.PHASE is used for processing, and not function package ARXEFSO.

General REXX Tips

REXX Samples

If you are interested in REXX/VSE samples, see the REXX/VSE homepage on

`http://www.ibm.com/systems/z/os/zvse/products/cf.html#rexx`

REXX Functions

In the VSE/REXX Reference Manual the correct syntax to invoke a REXX function is not specified with every function described. Nevertheless the syntax rules for REXX function apply, thus it is not possible to invoke a REXX function like a REXX command, for instance

```
ASSGN(STDOUT,SYSLOG)
```

Such a command is usually unknown to REXX, and the following error message is given:

```
ARX0565I LIBRM ERROR STATE RC=(0008,0000) ARXRX24 NO MATCH FOUND  ASSGN
```

REXX function calls are treated as expressions within a REXX statement. They are coded as

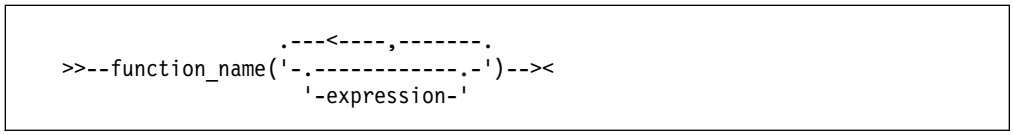


Figure 95. Syntax of a Function Call in REXX - Expression

Sample: oldout = ASSGN('STDOUT','SYSLOG')

Another valid syntax for function calls and subroutine calls is

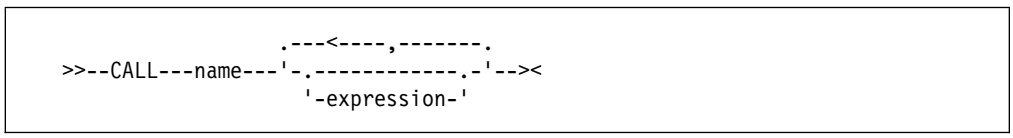


Figure 96. Syntax of a Function Call in REXX - Call

Sample: CALL ASSGN 'STDOUT','SYSLOG'

For further information see the REXX/VSE Reference manual, Chapter 3 "Keyword Instructions", "CALL".

POWER Command Environment

Error Handling

Always check the contents of the OUTTRAP variable after issuing a POWER command from REXX.

This is even necessary if the return code from the POWER command is zero !

A return code of zero just means that the POWER command has been successfully transferred to POWER. POWER error information is written to the stem variable specified on the OUTTRAP function.

CLASS Operand for GETQE

POWER queue entries can only be retrieved using GETQE, if the correct class value is specified, otherwise error message "ARX0950E RC=0004,FDBK=0001 received from VSE/POWER command" is returned. If you use the GETQE command without specifying operand CLASS, the default class A is used.

POWER Authorization

Accessing existing queue entries requires to specify a user ID and/or password. Accessing means either a GETQE or a POWER command like PALTER. Use the SETUID command to set the corresponding user ID (and password, if necessary). You can specify the FROM or TO user ID. Entries without a defined FROM/TO user ID can only be accessed, if an installation specific POWER master password has been defined in the system (see MPWD operand within skeleton SKPWRGEN in ICCF library 59, phase IPWPOWER). It provides access to all queue entries even if the user ID does not match.

Using the CONSOLE command environment, a TO-user may be set for an existing queue entry, like in the following sample:

```
fc = SENDCMD('PALTER LST,ENTRYX,USER=XXXX')
                                          /* set TO-user    */
'ADDRESS POWER'                /* switch command environment */
'SETUID XXXX'                             /* set POWER user */
'GETQE LST JOBNAME ENTRYX ...'            /* read entry     */
```

Figure 97. TO-user setting with REXX

Some of the POWER commands are only accepted for authorized user via the spool-access interface used by REXX/VSE: PFLUSH, PGO, PLOAD, PRESTART, PSETUP, PSTART, PSTOP, PVARY, PXMIT (see list of commands for PWRSPL Macro in the VSE/POWER Application Programming manual). If REXX/VSE users want to include those commands into their REXX procedure, the POWER master password should be defined during startup of POWER.

The REXX/VSE users specify this POWER master password in a SETUID statement, before issuing the POWER command. Code

"SETUID blabla mpwd"

in this case, where "blabla" is some non-blank string and "mpwd" is the defined POWER master password.

JCL Command Environment

DLBL Statements

REXX/VSE offers the ADDRESS JCL environment to imbed JCL commands. One JCL statement is processed after the other. This may cause a problem for DLBL statements, because JCL looks for further EXTENT statements before it updates the label area. If a REXX procedure issues a DLBL statement without any other JCL statement following, this DLBL is not activated!

If there is no further JCL statement following the DLBL statement within a REXX procedure, insert a dummy "/*" statement after the DLBL statement as shown in the manual REXX/VSE REFERENCE in chapter JCL Command Environment. JCL recognizes this as end condition for the DLBL statement and updates the label area also for the last DLBL statement.

SETPARM Statements

It is possible to set a symbolic parameter within a REXX procedure. If its value should be active for the higher Job Control Level on return, the same methods of addressing symbolic parameters apply as for JCL procedures (see EXEC statement in the manual "System Control Statements").

Here are samples to demonstrate the usage of symbolic parameters:

```
// JOB TSTSETP1
ACC S=DEVLIB.USCH
CAT TSTSETP1.PROC  R=Y
/* rexx procedure */
ARG TESTVAR
ADDRESS JCL "// SETPARM XXXX=" || TESTVAR
EXIT rc
/+
/*
// LIBDEF *,SEARCH=(DEVLIB.USCH,PRD1.BASE)
// SETPARM XXXX=' '
*                                  look here: XXXX is mentioned!
// EXEC REXX=TSTSETP1,PARM='UP',XXXX
/*
// IF XXXX = 'UP' THEN
// GOTO PARMUP
// IF XXXX = 'DOWN' THEN
// GOTO PARMDOWN
/. PARMUP
* PARM IS UP
// GOTO $EOJ
/. PARMDOWN
* PARM IS DOWN
/*
/&
```

Figure 98. Sample 1: Symbolic parameters

```
// JOB TSTSETP2
ACC S=DEVLIB.USCH
CAT TSTSETP2.PROC  R=Y
/* rexx procedure */
old = OUTTRAP(map.,,'NOCONCAT')
ADDRESS JCL "MAP &PARTIT"
do i=1 to map.0
  say map.i
end
EXIT rc
/+
/*
// LIBDEF *,SEARCH=(DEVLIB.USCH,PRD1.BASE)
// EXEC REXX=TSTSETP2,PARTIT=BG
/*
/&
```

Figure 99. Sample 2: Symbolic parameters

```
// JOB TSTSETP3
ACC S=DEVLIB.USCH
CAT TSTSETP3.PROC  R=Y
/* rexx procedure */
old = OUTTRAP(map.,,'NOCONCAT')
ADDRESS JCL "MAP &REXXVAR"
do i=1 to map.0
  say map.i
end
ADDRESS JCL "// SETPARM YYYY='NOT_SEEN'"
EXIT rc
/+
/*
// LIBDEF *,SEARCH=(DEVLIB.USCH,PRD1.BASE)
// SETPARM YYYY=F1
// EXEC REXX=TESTSETP,REXXVAR=&YYYY
/*
// IF YYYY = 'F1' THEN
// GOTO PARMF1
// IF YYYY = 'NOT_SEEN'
// GOTO PARMNS
// GOTO $EOJ
/. PARMF1
* PARM IS F1, WHICH IS AS EXPECTED
// GOTO $EOJ
/. PARMNS
* ??? PARM IS NOT_SEEN, WHICH IS NOT EXPECTED ???
/*
/&
```

Figure 100. Sample 3: Symbolic parameters

LINK Command Environment

Table of Authorized Programs ARXEOJTB

Programs called from REXX via ADDRESS LINK can be added to the Table of Authorized Programs ARXEOJTB as described in the REXX/VSE Reference Manual in Chapter 13. This is especially necessary,

- if the called program uses the EOJ macro,
- if the called program is to be loaded into the program area via LOAD (instead of GETVIS storage via CDLOAD),
- if the REXX program containing ADDRESS LINK invocations does not terminate successfully.

Console Command Environment

Return Codes From Macros MCSOPER, MCSOPMSG, MGCRE

The implementation of REXX console commands and functions is based on a privileged macro interface consisting of macros called MCSOPER, MCSOPMSG, and MGCRE. If problems occur, return and reason codes of these macros are displayed in message ARX0565I. A REXX program can also access and check return and reason code by using function SYSVAR together with parameter 'SYSERRCODES'.

Return and Reason Codes are returned in decimal format within message ARX0565I as well as with function SYSVAR.

They are explained in the manual *REXX/VSE Reference* or in the older manual *REXX/VSE Console Automation.*

Suspended Console

When using function GETMSG, execution may fail with function code 8 and the message

```
ARX0565I MCSOPMSG FAILED RC=(0012,0000)
```

meaning that the console is suspended. This happens if at least one message is queued for this console, but it is not retrieved within the next 15 seconds after its arrival in the message queue for this console.

Another reason could be that the REXX console program creates many messages while its REXX console is active. If more messages are generated than the z/VSE Console Router can handle, the console is suspended. Or to be more specific, if more than 12 unretrieved console messages are queued for a console, this console is suspended.

Whenever possible, this situation should be avoided by following the hints and tips given in the manual REXX/VSE Reference in chapter REXX/VSE Console Automation.

Check the priority of the partition running the REXX Console program !

If a console is suspended, messages arriving during the suspended time are lost for this console. They are put on the hardcopy file, which may also be processed by a REXX console program invoking the REDISPLAY command.

A REXX program can react on a suspended console by deactivating this console and activating it again.

Here is a code sample:

```
fc = GETMSG(msg.,'MSG',,,15)
If fc > 7 Then
Do
  Call SYSVAR 'syserrcodes'
  If syserrcodes = '0012 0000' Then
  Do
    'DEACTIVATE MYCONS'
    'ACTIVATE NAME MYCONS PROFILE REXALLRC'
  End
End
```

Figure 101. Sample: Reactivating a suspended console

When using REXX/VSE function FINDMSG and console is suspended, messages

```
ARX0565I MCSOPMSG FAILED RC(0012,0000)
ARX0960E ERROR RUNNING FUNCTION FINDMSG, RC=8
ARX0040I ERROR RUNNING <name>, line <n> : Incorrect call to routine.
```

occur. Then the REXX procedure abends. Using the following hint the REXX procedure may continue, thus allowing to deactivate and activate the suspended console. Specify

SIGNAL ON SYNTAX NAME <label>

before invocation of FINDMSG. This definition can be deleted after invocation of FINDMSG with

SIGNAL OFF SYNTAX

At label <label> the DEACTIVATE and ACTIVATE can be placed followed by a branch back to FINDMSG. Thus, message ARX0040I does not show up anymore and the REXX interpreter continues to execute the REXX procedure.

Retrieval of CICS Messages

If a REXX console is activated and a CICS console command is issued, there is no response when invoking function GETMSG with type 'RESP'. The only way to receive CICS answers on a REXX console is via function GETMSG with type 'MSG'.

Tracking of Operator Communication

Responses to operator-given commands are only routed to the console the operator is using for his communication, not to other defined master consoles. Thus it is not possible to track ongoing operator communications via a REXX console program. Since operator dialogs are logged on the hardcopy file, they can be scanned using the REDISPLAY command.

Function SENDCMD

Invocation of function SENDCMD in REXX/VSE may return message

```
ARX0565I MGCRE FAILED, RC=(0008,0018).
```

Implementation of function SENDCMD uses a dummy console. If another program running in the system uses this dummy console too, macro MGCRE returns with return code 8 and reason code 18 (X'12'). You can either handle this situation by some kind of retry-loop or avoid this situation by using a dedicated REXX console. Such a console needs to be activated first. Afterwards you can issue the command via 'ADDRESS CONSOLE' as is without using function SENDCMD.

Here is a code sample that invokes SENDCMD in a subroutine with automatic retry-loop:

```
Send_Console_Command: Procedure
Arg command
fc = 0
oldmsg = REXXMSG('OFF')
Do Until fc = 0
  fc = SENDCMD(command)             /* issue console cmd    */
  If fc /= 0
  Then Do
    Call SYSVAR 'syserrcodes'       /* determine error code */
    If syserrcodes = '0008 0018'    /* expected error       */
    Then Call Sleep 1               /* wait 1 second        */
    Else Return -1                  /* unexpected error     */
  End
End
msg = REXXMSG(oldmsg)
Return 0
```

Figure 102. Sample: SENDCMD with retry-loop

REXX SOCKET Function

Determine SOCKET Implementation

Since there exist two implementations of the REXX/VSE SOCKET function, the implementation active at your installation must be determined first in case of a problem. The different setups are described in “Setup for Using the REXX/VSE SOCKET Function” on page 217. Error message ARX0960E is caused by the IBM-provided SOCKET implementation, error messages SOC... indicate that the CSI-provided SOCKET implementation is used.

Debugging SOCKET Problems

The REXX/VSE Socket function is implemented using the socket calls offered by LE/C, which are again based on the BSD/C Sockets of TCP/IP. Complex problems of Socket programs require debugging on the lowest socket level: Rename member *XSOCKDBG.PHASE* in PRD2.TCPIPC to *$SOCKDBG.PHASE* and check the console output.

REXX and Other z/VSE components

REXX and ICCF

Even though there exists no special ICCF-support in REXX, you can do the following:

1. Invoke DTSUTIL via ADDRESS LINK (include DTSUTIL into ARXEOJTB first!).
2. With VSE/ESA 2.7, the REXX procedure ICCFTOOL.PROC is available in library IJSYSRS.SYSLIB. It does a global change of a given string within an ICCF library.
3. Read an ICCF member, by including it into SYSIPT via * $$ SLI and using either command EXECIO * DISKR SYSIPT or command PULL to store the member into a REXX stem variable.

```
// JOB TESTSLI
// EXEC LIBR
ACC S=PRIMARY.USCH
CAT TESTSLI.PROC  R=Y
/* rexx procedure */
'EXECIO * DISKR SYSIPT ( STEM ipt.'
...  /* further processing of ICCF-member within stem ipt. */
EXIT 0
/+
/*
// LIBDEF *,SEARCH=(PRIMARY.USCH,PRD1.BASE)
// EXEC REXX=TESTSLI
* $$ SLI ICCF=(OVERLAY2),LIB=(45)
/&
```

Figure 103. Sample: Read an ICCF member via SLI

4. At least some of the REXX procedures run in an ICCF interactive partition. Here is a small REXX procedure:

```
// JOB REXXICCF
// EXEC LIBR
ACC S=PRD1.BASE
CAT REXXICCF.PROC  R=Y
/* rexx procedure */
/* to be invoked from ICCF */
old = ASSGN(STDOUT,SYSLOG)
say 'hello world'
exit 0
/+
/*
/&
* $$ EOJ
```

Figure 104. Sample: REXX in an interactive partition - REXX procedure

And here is the corresponding ICCF Macro that can be invoked from an ICCF interactive partition:

```
* $$ JOB JNM=DTSUTIL ,DISP=D,CLASS=0,LDEST=(,SRVPROM5)
// JOB DTSUTIL
// ASSGN SYS010,DISK,VOL=CTS220,SHR
* DISCONNECT DTSFILE
// PAUSE
// EXEC DTSUTIL
PURGE LIB(45) MEMBER(REXXICCF)
ADD MEMBER(45,REXXICCF,USCH)
@MACRO
/SET CLASS I NOWAIT
/SET DELAY BYPASS
/INPUT
/LOAD ARXJCL PARM='REXXICCF'
/OPTION GETVIS=900
/END
/RUN
END OF MEMBER
/*
/&
* $$ EOJ
```

Figure 105. Sample: REXX in an interactive partition - ICCF macro

REXX and VSAM

With VSE/ESA 2.6 a new REXX/VSE command VSAMIO is offered to provide access to VSAM data. If you are still running on an VSE/ESA release prior to 2.6, you can do the following:

1. Invoke IDCAMS via ADDRESS LINK. This allows for instance to read a VSAM file using IDCAMS PRINT. The IDCAMS output is stored into the stem defined with the OUTTRAP function.
2. Make use of the DITTO-REXX-Interface to write and read a VSAM file. DITTO offers extra commands to copy data from tape, library member or VSAM file to a REXX stem variable and vice versa. DITTO command $VX allows to read a VSAM file into a stem and DITTO command $XV allows to write a VSAM file from a given stem.

```
call 'DITSETUP'   /* establish DITTO command environment */
ADDRESS DITTO     /* switch to the DITTO command environment */
/* read file into stem vsam. */
'DITTO $VX DSNIN=VSE.MESSAGE.ROUTING.FILE UCAT=VSAM.MASTER.CATALOG' ,
      'VARNAME=vsam.'
```

Figure 106. Sample: DITTO-REXX interface for VSAM

Chapter 11. Tape Library Support

The IBM Tape Library can be supported via LBSERV macro calls or JCL/AR LIBSERV statements. These commands are passed

- directly to the IBM Tape Library by VSE TLS (tape library support)
- via VGS (VSE Guest Server), when running VSE under VM

TS7700 logical volume sizes

As z/VSE does not support policy management the user has to define the logical volumes at the TS7700 Management Interface.
Please refer to the *TS7700 Virtualization Engine redbook* chapter 3 *Preinstallation planning and sizing* and for the definition of the logical volumes to chapter *4.3.8 TS7700 Virtualization Engine definition with the management interface* .

You can use the following link for the redbook.
http://www.redbooks.ibm.com/redbooks/pdfs/sg247975.pdf

Summary of possible return and reason codes

You will find the possible return and reason codes in the the manual *z/VSE 6.2 System Macros Reference* available at *http://publibfp.dhe.ibm.com/epubs/pdf/iesmfe81.pdf* .

The return and reason codes are listed in the following tables:

- *Table 23 Common Return and Reason Codes from LCDD, DFSMS/VM RMS, and VSE TLS Support*
- *Table 24 Additional Reason Codes Generated by LCDD and VSE TLS Support*
- *Table 25 Additional Reason Codes Generated by DFSMS/VM RMS*
- *Table 26: Reason Codes Generated by VGS*
- *Table 27 Reason Codes Generated by z/VSE* .

See also:

- *Table 45 MAPXPCCB Return Codes IJBXRETC*
- *Table 46 MAPXPCCB Reason Codes IJBXREAS* .

Chapter 12. Capacity Measurement Tool

The Capacity Measurement Tool (CMT) is shipped as part of z/VSE V4 and later. When CMT is started, SCRT89 accounting records are created and filed automatically in CMT datasets on z/VSE every four hours. The user can then extract records for the reporting period on z/VSE. Based on these records the sub-capacity report is created using SCRT.

Starting October 2017, a new Java version of SCRT must be used.

The following skeletons are provided:

SKCMT to prepare and activate the capacity measurrement tool

SKCMTINI to initialize the files used by the capacity measurrement tool

SKCMTREP to extract the records written by the capacity measurrement tool

Please refer to
http://www.ibm.com/systems/z/swprice/subcap/zvse.html
and
ftp://public.dhe.ibm.com/s390/zos/vse/pdf3/
Using_SCRT_with_zVSE_Best_Practices.pdf

If you want to use CMT on a z196, z114, zEC12, zBC12, z13, z13s, or z14, you need the following APAR fixes:

- DY47111 for z/VSE 4.2
- DY47110 for z/VSE 4.1

Refer to the z/VSE status page
http://www.ibm.com/systems/z/os/zvse/about/status.html for more information.

Chapter 13. Language Environment for VSE (LE/VSE)

LE/VSE Big Picture

Here is a brief discussion of LE/VSE release history.

- While LE/VSE 1.4.0 introduced many changes and enhancements in different product areas.
- Its follow-on LE/VSE 1.4.1 (in VSE/ESA 2.5) focused on structural improvements.
- LE/VSE 1.4.2 (in VSE/ESA 2.6) concentrated on dump facilities and optimization of storage tasks.
- LE/VSE 1.4.3 (in VSE/ESA 2.7) emphasized on new macro support, via CEELOPT, CEEFETCH and CEERELES and improvements to existing pre-initialization facilities and LE/CICS run-time option maintenance (new "CLER" transaction).
- LE/VSE 1.4.4 (in z/VSE 3.1) added new AR commands for LE/VSE installation status as well as further enhancements to callable services (CEE5TSTG), abend handling (messages, abend codes), existing assembler macro and dump support.
- LE/VSE 1.4.5 (in z/VSE 4.1) introduced new Callable Services to extract system information (CEE5INF), manage enclave environment (CEEENV) and suspend processing (CEEDLYM). Furthermore the AR-command interface was enhanced to allow for displaying LE/CICS Run-Time Options status via D CEE,CEECOPT command. DOS/PL1 Storage Compatibility (CLEAR parameter in STORAGE run-time option), new C-Run-Time Library Functions (strcasecmp and strncasecmp) and HEAPCHK run-time option support via the supplied CLER Transaction complemented the set of enhancements.
- With LE/VSE 1.4.6 (in z/VSE 4.2) additional Attention Routine (AR) commands were introduced to allow administrator/operator to even manage LE/VSE batch run-time options overrides directly from the console. Beyond this the LE PL/I runtime support has been improved to reduce limitations associated with the use of "fetch" for PL/1 application programs. As an optional extra the CEETRACE tracing tool was provided on an "AS-IS" basis (IBM does not officially support this feature). Irrespective of this it is designed to complement the LE/VSE dump and assist in application problem analysis by providing an execution-statement history. A COBOL/VSE source-code extraction is supplied with it, too.
- With LE/VSE 1.4.7 in z/VSE 4.3 a new Callable Service CEE5MC2 was supplied to return the default currency symbol for specified country code (either default or international currency symbol. From a programming point of view the CEEFETCH macro has been enhanced to support a wider range of possible target/candidate routines including new parameters (ENTRYPT and FTCHINFO). Furthermore the PL/I multi-tasking capability was added. When available hardware-wise, Break Event Address Register (BEAR) info about bad branches is included in CEE5DMP condition information section. Finally a new TCP/IP multiplexer configuration was introduced relying on a table that assigns the name of an interface phase to an unique SYSID value.

- With LE/VSE 1.4.8 in z/VSE 5.1 enhanced support for LE/C applications (e.g. VSAM SHR(4) file exploitation) and significant updates to System Programmer C (SPC) Environment and samples were provided. Furthermore the LE/VSE C Run-Time Socket API was enabled to support IPv6 (set of new functions). For PL/I users a new Callable Service CEEPUSR was made available to reserve programmer managed storage area in PL/I Multitasking environment (purpose of inter-task communication). For COBOL a new sample (IGZT5INF.C) for existent CEE5INF Callable Service was added. Finally the trace capabilities in optional extra CEETRACE tooling was enhanced with new functionality (mainly: *"Auto-Report Feature"*, *"High Level Language (HLL) Statement Exit"* and the *"Mini Dump"* option).

Changes with modification level LE/VSE 1.4.9 in z/VSE 5.2 and 6.1:

- Run-Unit Work Area (RUWA) Trace Control for CICS LE-Enabled Programs. A new transaction is supplied with LE/VSE 1.4.9 (called "CRUT") which can be used to activate, query and deactivate the run-unit work-area (RUWA) tracing function. This capability provides information on a LE/VSE CICS program's run-unit storage requirements. Specifically it includes both application and run-time storage amounts needed to successfully execute the program. Documentation can be found in the LE/VSE Debugging Guide and Run-Time Messages" in the section titled "CICS Run-Unit Work-Area Storage Tracing.
- Enhanced CEMT I PROG CICS command capability to show language "Pli" for a fetched PL/1 subroutine. The first load or fetch of a target PL/1 subroutine will now allow LE/VSE to provide CICS with the necessary information (customer requirement).
- Better performance for iconv() open functionality when using UCS-2 table converters (customer requirement).
- A new Attention Routine command (S CEE,REFRESH) was made available to allow for instant update of LE/VSE status information (e.g. in case PTFs have been applied).
- Batch storage requirements for ALL31(ON) environments have been further reduced with LE/VSE 1.4.9.
- The NEWC transaction is now capable not only reloading LE/CICS run-time options but also refreshing the CEETRACE interface routine if currently active in the CICS system.
- New C/VSE sample (CELTVIVP.Z) for CEETRACE optional install verification.

Changes with modification level LE/VSE 1.4.10 in z/VSE 6.2:

- Further enhanced support for LE/VSE conforming Assembler Main. A new CICS translator option LEASM was introducted that instructs the CICS translator to generate appropriate entry and exit coding to make an assembler program an LE/VSE conforming MAIN. This also goes along with the possibility for more comprehensive condition handling in an LE/VSE conforming Assembler main routine (by use of EXEC CICS HANDLE CONDITION and EXEC CICS HANDLE ABEND LABEL).
- Miscellaneous improvements to LE/VSE sample programs (all languages) as supplied in installation sublibrary PRD2.SCEEBASE.

For more details on release/mod/level specific contents please refer to z/VSE home, pdf-document "Recent enhancements for LE/VSE".

http://www.ibm.com/systems/z/os/zvse/products/languages.html#le

The following sections are intended to ease product usage, customization and problem follow-up.

See also Chapter 22, “Debug Tool for VSE/ESA (LE)” on page 331 or CEETRACE on page 241 for additional debugging aids.

LE/VSE 1.4.9 in z/VSE 5.2

The following table lists the component identifiers (COMP IDs) and component level codes (CLCs) relevant for LE/VSE 1.4.9.

Component-ID	CLC	Description
5686-CF9-32	52K	LE Common base, containing information written in: - Uppercase and mixed-case US English - Japanese NLF
5686-CF9-33	52L	LE C-specific base, containing information written in: - Uppercase and mixed-case US English - Japanese NLF
5686-CF9-34	52M	Optional LE DBCS Locale Component (see note 2)
5686-CF9-36	52W	LE COBOL-specific base and CICS, containing information written in: - Uppercase and mixed-case US English - Japanese NLF
5686-CF9-37	52Z	LE PL/I-specific base, containing information written in: - Uppercase and mixed-case US English - Japanese NLF

Notes:

1. The LE Base ($SVACEE) and C Runtime ($SVAEDCM) are pre loaded in the SVA. Please also see the *z/VSE Planning Guide*, SC34-2681.

   Please notice that option modules (for example) CEECOPT.PHASE and CEEDOPT.PHASE are included in SVA loadlist $SVACEE. Changing run-time options for batch and CICS environment therefore requires a SVA re-load via SET SDL command from BG partition. Supplied skeletons CEEWCOPT and CEEWDOPT in ICCF library 62 will take this into consideration.
2. The optional LE/VSE 1.4.9 DBCS locale component is shipped on the z/VSE 5.2 Extended Base tape.
3. For summary references about LE/VSE, please see the z/VSE Release Guide SC34-2696 which is also available on the z/VSE Softcopy Collection Kit SK3T-8348.

Also see *IBM LE/VSE Customization Guide*, SC33-6682 for related information.

LE/VSE 1.4.9 in z/VSE 6.1

The following table lists the component identifiers (COMP IDs) and component level codes (CLCs) relevant for LE/VSE 1.4.9.

Component-ID	CLC	Description
5686-VS6-32	61K	LE Common base, containing information written in: - Uppercase and mixed-case US English - Japanese NLF
5686-VS6-33	61L	LE C-specific base, containing information written in: - Uppercase and mixed-case US English - Japanese NLF
5686-VS6-34	61M	Optional LE DBCS Locale Component (see note 2)
5686-VS6-36	61W	LE COBOL-specific base and CICS, containing information written in: - Uppercase and mixed-case US English - Japanese NLF
5686-VS6-37	61Z	LE PL/I-specific base, containing information written in: - Uppercase and mixed-case US English - Japanese NLF

Notes:

1. The LE Base ($SVACEE) and C Runtime ($SVAEDCM) are pre loaded in the SVA. Please also see the *z/VSE Planning Guide*, SC34-2681.

   Please notice that option modules (for example) CEECOPT.PHASE and CEEDOPT.PHASE are included in SVA loadlist $SVACEE. Changing run-time options for batch and CICS environment therefore requires a SVA re-load via SET SDL command from BG partition. Supplied skeletons CEEWCOPT and CEEWDOPT in ICCF library 62 will take this into consideration.

2. The optional LE/VSE 1.4.9 DBCS locale component is shipped on the z/VSE 6.1 Extended Base tape.

LE/VSE 1.4.10 in z/VSE 6.2

The following table lists the component identifiers (COMP IDs) and component level codes (CLCs) relevant for LE/VSE 1.4.10.

Component-ID	CLC	Description
5686-VS6-32	62K	LE Common base, containing information written in: - Uppercase and mixed-case US English - Japanese NLF
5686-VS6-33	62L	LE C-specific base, containing information written in: - Uppercase and mixed-case US English - Japanese NLF
5686-VS6-34	62M	Optional LE DBCS Locale Component (see note 2)
5686-VS6-36	62W	LE COBOL-specific base and CICS, containing information written in: - Uppercase and mixed-case US English - Japanese NLF
5686-VS6-37	62Z	LE PL/I-specific base, containing information written in: - Uppercase and mixed-case US English - Japanese NLF

Notes:

1. The LE Base ($SVACEE) and C Runtime ($SVAEDCM) are pre loaded in the SVA. Please also see the *z/VSE Planning Guide*, SC34-2681.

   Please notice that option modules (for example) CEECOPT.PHASE and CEEDOPT.PHASE are included in SVA loadlist $SVACEE. Changing run-time options for batch and CICS environment therefore requires a SVA re-load via SET SDL command from BG partition. Supplied skeletons CEEWCOPT and CEEWDOPT in ICCF library 62 will take this into consideration.

2. The optional LE/VSE 1.4.10 DBCS locale component is shipped on the z/VSE 6.2 Extended Base tape.

LE/VSE Attention Routine Interface and Commands

The attention routine interface is pre-customised and activated in z/VSE. It is recommended to keep this interface enabled since it suits to display LE/VSE run-time option, exit and status reports. In this context please ensure system ASI procedure USERBG is current and contains the following VSE POWER statement:

```
// PWR PRELEASE RDR,CEEWARC              LE - AR INTERFACE
```

This particularily applies if there is an equivalent or previously tailored version of this procedure in place. For the system supplied version of USERBG.PROC, please refer to skeleton SKUSERBG in ICCF library 59. Finally please make sure job CEEWARC is preloaded in VSE POWER RDR queue. In case of doubt or need to activate please refer to skeleton CEEWARC in ICCF library 62.

Furthermore the AR interface also accepts commands that will allow to manage LE/VSE batch run-time option overrides. For details and reference on such commands please see the LE/VSE Debugging Guide and Run-Time Messages, SC33-6681, section "Using Attention Routine Interface Commands".

LE/VSE Run-time Options

From z/VSE 3.1 onwards, LE/VSE checks if an invalid (older) run-time option module is loaded. In this case ABEND U4093 RSN42 is issued (for batch), respectively LE/VSE return code 11060 (at CICS initialisation time). In particular, it is recommended to save customized versions of LE/VSE batch and CICS run-time option sources - prior to performing an FSU - if stored in LE/VSE product library PRD2.SCEEBASE. Should there be a need to re-establish option changes of this kind, please use current job skeletons CEEWDOPT (batch) or CEEWCOPT (CICS) as supplied in ICCF library 62. There is no need at all for the above task, if default LE/VSE or CICS run-option modules are already used (as supplied with the z/VSE release installed).

Mixed Language Applications under LE/VSE (involving Assembler)

When creating or maintaining mixed language applications in an LE/VSE environment various supported techniques are available. In general the following macros and services can assist to ensure operating LE/VSE-conform.

- LE/VSE assembler macros (CEEENTRY/CEETERM)
- LE/VSE preinitialization service (CEEPIPI)
- LE/VSE C-specific macros (EDCPRLG/EDCEPIL)

Examples and further details not covered here are available on z/VSE home via:

`http://www.ibm.com/systems/z/os/zvse/downloads/samples.html#le`

Summary of LE/VSE Customization and Verification Jobs

A list of LE/VSE related jobs, pre-installed in ICCF library 62 is maintained in *LE/VSE Customization Guide*, SC33-6682, section 1.10 IBM-Provided Customization and Verification Jobs. The associated link will be as follows: http://publibz.boulder.ibm.com/cgi-bin/bookmgr/BOOKS/fl2ice07/1.10

The members referenced herein can assist in various verification and customization tasks.

Languages and CICS Transaction Server

CICS Transaction Server for z/VSE supports:

- All LE/VSE conforming compilers (C/VSE, COBOL/VSE, PLI/VSE)
- High Level Assembler with LE/VSE
- DOS/VS COBOL and VS COBOL-II (if relinked / using LE run-time)

CICS Transaction Server will not support:

- DOS PL/1 and C/370 applications (recompilation with LE-conforming compiler is required)
- RPG-II (not supported by LE, either)

Note: Be aware that every CICS TS related transaction needs to be "security-enabled" prior to it's first execution. The Interactive Interfaces associated support via dialog "Security Maintenance (fast path 28).

Generating Applications Capable of Running Under LE/VSE

Remember Interactive Interface capabilities when building your own applications (see Primary Library, OPTION 8, translate and compile).

Not only does it provides applicable JCL-coding, but it also ensures involving language independent stubs as well as useful compile options.

However there are further issues such as the common question of how to implement "application-specific" option overrides in a program. This is easy to realize based on JCL skeletons provided. Here is a short example:

```
// JOB COMPILE
// etc
// OPTION CATAL
   PHASE name,*
   INCLUDE DFHELII  (language independent stub, only if CICS)
   INCLUDE CEEUOPT  (the common way to include application specific overrides)
   INCLUDE MYOPTS,(CEEUOPT)  (alternate way with user name + CSECT reference!)
// EXEC IGYCRCTL    (compiler invocation)
  (source code goes here) etc...
```

Note: *LE/VSE Customization Guide*, SC33-6682, Chapter 2 shows an example on how to build a CEEUOPT.OBJ (application specific option module). Please also notice that the catalogued object name used is open, however never change the CSECT name used for option module generation. Finally, in case product related run-time options may change there might be a need to update old or obsolete application specific settings.

PL/1 for VSE/ESA or C for VSE/ESA users should consider using the "PLIXOPT" or "#pragma runopts" definitions to create option over-rides in their "main" routines instead of a CEEUOPT.

Overriding Run-Time Options with an EXEC card (Batch)

There may arise the need to override installation-wide run-time options (temporarily) while running a single application. For example if a dump needs to be taken. For batch you may wish to pass the desired run-time options via PARM-string definition in the EXEC statement. Then the following should be considered:

- The "/" character is used as a separator between program-arguments and run-time options.
- The position of the slash is related to the setting of run-time option CBLOPTS.
- Presuming a default setting of CBLOPTS(ON) there is a language dependent approach as follows.
  - C or PL/I: A trailing slash after the run-time options is required.

    ```
    // EXEC PGM,SIZE=PGM,PARM='RPTSTG(ON)/program arguments if any'
    ```

    This example overrides the default run-time option RPTSTG(OFF) producing a storage report to the current job output.
  - COBOL: Program arguments are expected before run-time options. Therefore the slash should be specified in a way like

    ```
    // EXEC PGM,SIZE=PGM,PARM='program arguments if any/RPTOPTS(ON)'
    ```

This example overrides the default run-time option RPTOPTS(OFF) producing a run-time option report to the current job output.

– This difference mainly exists to support the way programmers are accustomed to interact with programs written in a specific language. A setting of CBLOPTS(OFF) would suit to support overrides in a common way (all languages).

Note: There are many related references in the LE/VSE Bookshelf, mainly in *LE/VSE Customization Guide*, SC33-6682, Appendix A + section "COBOL Compatibility", LE/VSE Programming Guide SC33-6685 Chapter 4,5 and LE/VSE Programming Reference SC33-6685 chapter 2. Non-LE conforming assembler programs that act as drivers for LE-conforming applications cannot, by default, pass any runtime option overrides supplied on the JCL PARM card.

AMODE 24 Applications in a LE/VSE-CICS Environment

Under CICS, the supplied default LE/VSE run-time options ALL31(ON) and STACK(4K,4080,ANYWHERE,KEEP) are present.

These settings improve performance and storage utilization for a CICS region running AMODE31/ANY programs or applications that use CICS services to invoke AMODE24 programs from AMODE31/ANY applications. This is applicable unless AMODE31/ANY programs dynamically call AMODE24 programs (which are not automatically enabled for AMODE switching). In such cases a setting off ALL31(OFF) and STACK(4K,4080,BELOW,KEEP) is required. You may wish to implement it by use of an appropriate CEEUOPT.OBJ linked with the application.

Detailed information about run-time option changes are available in *LE/VSE Customization Guide*, SC33-6682.

Useful LE/VSE Run-Time Options and Those to Use with Caution

There exists a pdf-document "Useful LE/VSE Run-Time Options" on z/VSE home that covers more details about frequently used run-time options via:
http://www.ibm.com/systems/z/os/zvse/products/languages.html#le or
ftp://public.dhe.ibm.com/eserver/zseries/zos/vse/pdf3/LE_Useful_Run_Time_Options.pdf

LE/CICS-wide Run-Time Options to VSE Console

You may wish to display active LE/CICS-wide run-time option defaults immediatelly (without the need on global report option setting via run-time option RPTOPTS(ON)).

This is supported via transaction "ROPC" (security enabled for CICS Transaction server) and shipped in CICS GROUP (CEE). Running this transaction will show LE/CICS-wide default options on VSE console.

Also available is transaction "NEWC" which allows for the LE CICS-wide run-time options to be "newcopied" while CICS is still running. This is useful if the CEECOPT.PHASE options module has been changed and the new settings are required to be activated without a restart of the CICS system.

CICS Translator Options Required for COBOL Applications

For COBOL/VSE (or VS COBOL II programs) to run under CICS one of the following CICS TS translator options should be used to prevent abends. This will apply to mainline programs as well as to COPY-books that may be translated separately.

VS COBOL II programs (if still involved nowadays) should be legacy, error-free running application code, built prior to end-of-service of this product. Please refer to COBOL/VSE Migration Guide, GC26-8070-00, for more information.

Apart of the above please also be aware of this

- XOPTS(**COBOL2**) was a minimum setting for VS COBOL II type programs, preferably matching the ANSI74 standard. It may however also be used in a COBOL/VSE context.
- XOPTS(**ANSI85**) implies COBOL2 and can also be applicable for COBOL/VSE or VS COBOL II compiled program units. However these settings are appropriate in case the application exploits ANSI85 functionality such as nested programs.
- XOPTS(**COBOL3**) is a CICS TS only translator option which you may wish to use as an indicator for a COBOL/VSE or SAA AD/Cycle COBOL/370 cross compile program unit. It implies ANSI85 and COBOL2. By the way, CICS TS doesn't support the SIT COBOL2 parameter since the run-time must be LE/VSE. Please refer to CICS Transaction Server documentation for more details.

Note: Sometimes VS COBOL II programs, translated without either of these CICS translator options, executed without error with the former VS COBOL II run-time. However such programs will not execute successfully under LE/VSE and are likely to use the old and obsolete COBOL specific CICS stub DFHECI.

Writing Language Environment Main Assembler Applications for CICS/TS

There exists a pdf-document on z/VSE home that covers more details about writing LE/VSE conforming assembler main applications for CICS/TS : *Writing Language Environment Main Assembler Applications for CICS/TS* or

`ftp://public.dhe.ibm.com/eserver/zseries/zos/vse/pdf3/LE_Assembler_main_under_CICS.pdf`

CLER Usage in multiple CICS environments

This LE/CICS run-time option tailoring transaction is intended for dynamic, temporary overrides in a single, active CICS TS subsystem. As such, CLER customization will not take effect in other CICS subsystems nor are the changes introduced retained after CICS recycling. However LE/VSE run-time options that are commonly shared between multiple CICS subsystems (referring to the same CEECOPT module) are better tailored via customizing CEEWCOPT skeleton and CEECOPT.A option source.

Common Abend Conditions With LE/VSE

LE/VSE Specific Abend Codes and Their Possible Causes

Some LE/VSE abend codes are referenced in LE error messages (beginning either with 'CEEnnnnn', 'EDCnnnnn', 'IBMnnnnn' or 'IGZnnnnn'), or via CICS error messages (beginning with 'DFHnnnnn').

Further details can be found either in *LE/VSE Debugging Guide and Run-Time Messages* , SC33-6681 or at the operator console by entering the string CEEABENDC in the command line and entering PF9.

There also exists a pdf-document "Common ABEND Conditions with LE/VSE" on z/VSE home providing guidelines in case of an LE/VSE abend event.
http://www.ibm.com/systems/z/os/zvse/support/le.html or
ftp://public.dhe.ibm.com/eserver/zseries/zos/vse/pdf3/LE_Common_ABEND_Conditions.pdf

Language Environment Code Set Conversion

There exists a pdf-document on z/VSE home that covers more details about

- Direct conversion and indirect conversion via Universal Character Set 2 (UCS-2).
- Conversion from or to UCS-2 and UTF-8 (USC Transformation Format 8).

Here is the link: *Language Environment Code Set Conversion*

`http://www.ibm.com/systems/z/os/zvse/products/languages.html#le`

or
ftp://public.dhe.ibm.com/eserver/zseries/zos/vse/pdf3/LE_Code_Set_Conversion.pdf

LE/VSE Related Service via Ordering PSP Bucket

Please use VSE specific PSP buckets for upgrading LE/VSE. These packages are named "ZVSEvrm" (where v is the z/VSE version, r the release and m the modification level). Appropriate versions can be ordered via IBM service teams.

Current subsets for LE/VSE are named:
"IBMLANG/62K" (z/VSE 6.2),
"IBMLANG/61K" (z/VSE 6.1),
"IBMLANG/52K" (z/VSE 5.2),
"IBMLANG/51K" (z/VSE 5.1)

Of further interest can be CICS subsets ensuring synchronisation with LE/VSE.

For latest "in service" z/VSE status please refer to z/VSE release status page on z/VSE home.

`http://www.ibm.com/systems/z/os/zvse/about/status.html`

LE/VSE Documentation Links

- The following link provides further information via IBM Knowledge Center's PDF-library.
  https://www.ibm.com/support/knowledgecenter/SSB27H_6.2.0/pdfs_6.2.0_neu.html

LE/VSE Download Tools

The following link provides information on LE/VSE specific tooling available for download on z/VSE Home.

`http://www.ibm.com/systems/z/os/zvse/downloads/tools.html`

- LE/VSE CEETRACE Feature: CEETRACE is not intended to replace the LE z/VSE dump information or the Debug Tool for VSE/ESA. Instead it is designed to complement the already available LE z/VSE dump information to aid in application problem analysis by providing an execution statement history prior to any subsequent application failure similar to the previously available READY TRACE facility of DOS/VS COBOL. Applications that do not abend will not automatically produce an execution statement report.
  Please see the PDF documentation inside the ZIP file for how to install and enable the CEETRACE Feature.

  `http://www.ibm.com/systems/z/os/zvse/downloads/tools.html#ceetrace`

- "LEVSE_Control_Center" (LECC) an all in-one package for common language tasks. It contains advisory pilot functionality and integrates, respectively interfaces to tooling listed below. LECC V4 is the recommended level to consider for use with z/VSE 5.2 and higher target system(s). Use with previous z/VSE target systems is not generally restricted however you may experience problems if trying to exploit functionality that is not available on older or unsupported z/VSE systems.

  `http://www.ibm.com/systems/z/os/zvse/downloads/tools.html#lecc`

  LECC actually comprises:

  1. "CEETRACE" interfacing. LECC just uses the CEETRACE feature and provides many explore options. This comprises installation, supplied install verification jobs, status/option reports and trace customization/setup.
  2. "JCalc_LEVSE Tool" : calculate size of LE/VSE eligible SVA modules
  3. "JLink_LEVSE_Analyzer" : validate VSE compile/link lists of existing programs, check options, stub/run-time routines, resources, programming techniques etc. in use and get a summary report in html-format
  4. "JRun_LEVSE_Samples" : run supplied language sample programs exploiting LE/VSE Callable Services (in different high-level languages).

  Note: For each tooling, referred above, there exists a direct invocation button in "LEVSE_Control_Center" (located on the main panel).

Chapter 14. z/VSE Security and Security Migration

From z/VSE 4.3 onwards, CICS/VSE is no longer shipped on the Extended Base Tape.

From z/VSE 5.1 onwards, CICS/VSE is not supported.

From October 31 2012 CICS/VSE is no longer supported on any z/VSE release.

If you need more information about CICS/VSE, please see a previous version of this document.

After initial installation the security concept based on the Basic Security Manager (BSM) is active.

In case of an FSU you can migrate your security definitions.

If you need more information on how to migrate security, please see a previous version of this document.

What to do if Problems occur

After Migration to the new Security Concept

After having migrated to the security concept based on the BSM there are a couple of things that may happen:

- The new dialogs (e.g. fastpath 285) do not work, here some possible reasons:
  - Selection panels and Application profiles were not upgraded
  - Transactions IETV and IETS are not defined, this is possibly because new security support was implemented by installing the PTF. Please run SKCSDFIL.
  - After the security is migrated to the concept based on the BSM in case above transactions are not authorized, following job will help:

```
// EXEC BSTADMIN
AD TCICSTRN IETS
PE TCICSTRN IETS ID(GROUP61) AC(READ)
AD TCICSTRN IETV
PE TCICSTRN IETV ID(GROUP61) AC(READ)
PF D R
/*
```

- CICS does not come up after migrating to the BSM based security concept:

```
DFHXS1104 DBDCCICS
  Default security could not be established for userid CICSUSER. The
  security domain cannot continue, so CICS is terminated. SAF codes are
  (X'00000008',X'00000000'). ESM codes are (X'00000034',X'00000000').
```

  You have to add the CICSUSER on the access list of this profile with READ authorization:

```
// EXEC BSTADMIN
PE APPL DBDCCICS ID(CICSUSER) AC(READ)
PF D R
/*
```

- User PROG or OPER are not authorized. Newly added users are not authorized. This is most likely because you forgot to press PF6 (Groups) in User Profile Maintenance (fastpath 211) to migrate user groups. PF6 will generate a job in the punch queue, you have to copy it to primary library and submit it.

B-Transient Considerations

B-Transients are a special kind of system phases.

In a system with batch security active (**SEC=YES**), B-transients can be:

- **loaded from protected libraries only**
- **cataloged into protected libraries only**

Libraries that contain B-transients of the z/VSE base programs are automatically protected by appropriate entries in the pregenerated access control table DTSECTAB.

For any other B-transients, the user has to define the LIBRARY containing the B-transients as PROTECTED.

In general, resources without an entry in DTSECTAB are NOT protected.

If an access violation occurs and the library was defined as protected, please check whether:

1. The library is in VSAM space or is a BAM library (this is shown by the output of the EXEC LIBR command LD L=libname O=ST).
   - For library in VSAM space an '*' has to be specified as VOLID

     DTSECTAB TYPE=LIB,**NAME=*.VSE.REP1.LIBRARY.REP1**,UACC=CON
   - For libraries managed by BAM, the required 1 to 6 character volume identifier on which the library resides has to be coded in DTSECTAB:

     DTSECTAB TYPE=LIB,**NAME=888888.P.C.TEST.LIB1**,UACC=CON
2. The B-transient phase exists more than once in the system. If that is the case, check if the LIBDEF chain points to a library which is NOT protected.

Array Overflow When Assembling DTSECTAB

If the number of entries in DTSECTAB is higher than the maximum defined in the macro source, two assembler messages (**ASMA254I**) like this will be shown:

```
4,ARRAY OVERFLOW, MAXIMUM TABLE SOURCE STATEMENTS ALLOWED IS 1001
4,CONTACT SYSTEM ADMINISTRATOR TO UPDATE DTSECTAB MAC SOURCE
```

The sample shows a limit of 1001 entries in the table DTSECTAB. This limit is specified in the macro source of DTSECTAB. At the beginning of the DTSECTAB macro sources there is a description of how to change this value.

1. Change all assignments of 1001 to your higher value (**maximum is 32767**). For example:

```
          GBLC  &C(1001)        **TYPE= TRANSLATED CODE(B,C,..)
          GBLC  &I(1001)        **SUBTYPE=
          GBLC  &L(1001)        **LOGGING OPTIONS
          GBLC  &N(1001)        **NAME=
          GBLC  &S(1001)        **ACCESS CLASSES
          GBLC  &T(1001)        **TYPE=
            ...
&MAX      SETA  1001            **NUMBER OF ARRAY ENTRIES**
```

2. When the following message appears, the ACTR values have to be increased as well.

   **** ASMA013S ACTR counter exceeded - DTSEC**

   Note: There might be more than one ACTR statement to be changed.

Skeletons for CICS Dumps

In ICCF library 59 the following skeletons are available.

Task	CICS TS
Print auxtrace file	DFHAUXPR
Print transaction ABEND information on dataset A/B, PRTDUxxx job in POWER reader.	SKDFHDUP PRTDUMPA PRTDUMPB
Info Ana print format dump	storage dump management dialog
Info Ana print trace out of dump	SKCIDTRA or storage dump management dialog

Jobs for CICS Dumps

The following jobs to handle CICS dumps are available in the POWER reader queue.

Task	CICS TS
Print dump dataset A of DBDCCICS	PRTDUMPA
Print dump dataset B of DBDCCICS	PRTDUMPB
Print dump dataset A of CICS2	PRTDUC2A
Print dump dataset B of CICS2	PRTDUC2B

Transaction Security

If you see the following message, the transaction may not be defined to the basic security manager.

Use the dialog *Security Maintenance* to add the definition.

```
DFHAC2033 09:41:23 DBDCCICS
     You are not authorized to use transaction XXXX.
     Check that the transaction name is correct.
```

Where Can I Get More Information about Security

The following table provides an overview of the available security documentation.

Figure 107. Security Documentation

Document	Description
z/VSE Planning	It describes z/VSE and provides information which helps you plan and setup z/VSE. Chapter: Security Support Provides an overview of the z/VSE security support. Chapter: System Authorization Facility (SAF) and its External Security Interface (RACROUTE) This is additional information to write your own security routines.
z/VSE Administration	Chapter: Protecting z/VSE Resources This is the main part of the security documentation. It contains a comprehensive description of the z/VSE security support.
OS/390 Security Server External Security Interface (RACROUTE) Macro Reference (GC28-1922)	Contains a description of the RACROUTE macro parameters which are required to code a security or a resource manager which issues security checks.
OS/390 Security Server (RACF) Data Areas (SY27-2640)	This book contains the description of data areas used by the RACROUTE macro. They are required to code a security or a resource manager which issues security checks.

Chapter 15. CICS TS for z/VSE 2.2

CICS Transaction Server for z/VSE 2.2 (CICS TS for z/VSE 2.2) replaces CICS Transaction Server for z/VSE 2.1 (CICS TS for z/VSE 2.1) and CICS Transaction Server for VSE/ESA 1.1.1 (CICS TS for VSE/ESA 1.1.1).
The functionality of CICS TS for VSE/ESA 1.1.1 and CICS TS for z/VSE 2.1 is contained in CICS TS for z/VSE 2.2.
CICS TS for z/VSE 2.2 is the only CICS version that can be used with z/VSE 6.2.

Hints for Implementing CICS TS for z/VSE V2.2

With CICS TS for z/VSE V2.2 all CICS TS tables have to be reassembled. It is strongly recommended to use the tables, provided in ICCF Lib 59 as skeletons for own customizing. Never use old tables from a previous CICS TS or CICS/VSE.
For new or changed parameters of DFHSIT refer to skeleton DFHSITSP or DFHSITC2.
DFHPPT and DFHPCT can no longer be used - DFHTCT can be used for non-VTAM-Terminals only (e.g. sequential terminals).

It is also recommended to use the VSELIST as base for GRPLIST and to hold all user GROUPs in a separate LIST. It is now possible to specify up to four LISTs in GRPLIST. VSELIST and up to three user LISTs can be specified in GRPLIST:

```
GRPLIST=(VSELIST,userlst1,userlst2,userlst3),              C
```

That means mixing of system and user definitions in only one list is no longer required.

And do not forget: use EXEC DFHSIP,SIZE=DFHSIP,....**,OS390**. ICCF member SKCICS in Lib 59 is a good starter.

VSE/ICCF runs with CICS TS only. This is true for the Interactive Interface too.

Version dependent functions

There are some functions which are named version dependent. These are

- DFHDU430
- DFHPD430
- DFHTT430
- DFHTU430

With z/VSE 6.2 you can run only these functions, not the older versions.

Error Message DFHPA1107 During CICS TS Startup

z/VSE 6.2 runs with CICS TS for z/VSE 2.2 only. If the phase of a configured DFHSPITSP or an own SIT DFHSITxx was restored from an old z/VSE system, you will see message

```
DFHPA1107 DBDCCICS A  420 VERSION OF MODULE DFHSITSP WAS LOADED.
CICS CAN ONLY INITIALIZE WITH THE CURRENT LEVEL SIT.
```

during CICS startup and the CICS startup fails.

In this case you have to recompile the SIT with CICS/TS 2.2.
A basic start can be used to bring up the system with a basic CICS.

CEMT I TASK

With CICS TS there are the following three states:

RUN for the task currently running - it's the CEMT task
DIS for tasks which are dispatchable - ready to run
SUS for tasks which are in any kind of a wait state

When you type a question mark in front of a listed task you get more information about the reason why e.g. that task is hold in a wait state. This extended display by typing a question mark is available for all CEMT INQUIRE commands. The following displays show an example:

Normal display- CEMT I TA

```
Tas(0000215) Tra(VSAM) Fac(A001) Sus Ter Pri( 001 )
   Sta(TO) Use(EBER    ) Rec(X'B54D3A486B4E3800') Hty(FCIOWAIT)
```

Extended display - put a question mark in front of the related task to be displayed in detail:

```
Task(0000215)
Tranid(VSAM)
Facility(A001)
Runstatus(Suspended)
Ftype(Term)
Priority( 001 )
Purgetype(               )
Startcode(TO)
Userid(EBER)
Recunitid(X'E77DC2F5F4C4F3C1')
Htype(FCIOWAIT)
Hvalue(FILEA)
Htime(000000)
Bridge()
Identifier()
```

where 'Hty' and 'Htype' mean 'Hold Type'.

In the previous example VSAM is waiting for completion of a VSAM I/O for FILEA. Hold time is below 1 second.

There are 155 Hold Types, all described in the 'Problem Determination Guide'. With this information it is now possible to determine the wait reason, wait type and wait time for a task without having a CICS monitor.

CEMT SET PROGRAM(progname) PHASEIN

With CICS TS there is a new newcopy command by saying:

```
CEMT SET PROGRAM(progname) PHASEIN
```

This program will be fetched into storage independent of its RESCOUNT.

All tasks currently using this program will run with the old copy, all new attached task will use the new copy of that program. The old copy will be invalidated after the last task using this copy has been finished.

CEDF Usability Enhanced

With CICS TS the CICS supplied transactions CECI and CEBR can now be invoked from a CEDF session.
This is done by going from a normal CEDF screen to WORKING STORAGE (PF5) and than pressing PF2 BROWSE TEMP STORAGE or PF5 INVOKE CECI.

CEDX Transaction Allows Debugging of Non-Terminal Tasks

Prior to CICS TS 1.1.1, non-terminal tasks couldn't be debugged with CEDF.

Starting with CICS TS 1.1.1 debugging of such tasks is available with the new CICS supplied transaction CEDX. This is initiated by typing:

```
CEDX xxxx      where xxxx stands for TRANID
```

This includes debugging of tasks started by CICS Web Support (CWS), which are handled as non-terminal tasks.

Priority Aging Set by PRTYAGE

Beside the possibility of specifying a terminal, transaction or operator priority for a task, there is a new way of task prioritization: priority aging. If a task is ready to run (seen as DISpatchable) the priority of this task is raised by 1 each time PRTYAGE elapsed.

This new way of changing the priority automatically is useful especially in production systems with transactions consuming a high amount of CPU resources. The initial value of PRTYAGE, delivered with DFHSITSP, is 5000 milliseconds. This value can be changed via CEMT INQ SYSTEM. Just overwrite the value shown right of 'AGING'. There is no general recommendation for this setting, but a value of 500 to 1000 seems to be more appropriate. It has to be tried out in each individual installation.

Loading of Resident Programs

In CICS/VSE it was possible, to load a program during CICS/VSE start up just by specifying system related parameters, e. g. via DFHALT entries for RMODE(24) programs or RES=YES for RMODE(ANY) programs.

This has been changed in CICS TS. Programs are loaded on demand only.
To load such programs during CICS TS startup it is recommended to start a PLTPI program which does a

```
EXEC CICS LOAD PROGRAM(progname) HOLD
```

This could be useful in case of huge programs by avoiding long fetch time on first reference.

Autoinstall-Exit for Programs: DFHPGADX

With CICS TS the exit-program DFHPGADX allows you to customize the definition of program entries to your needs. Candidates for changes of parameters in this exit are e.g. EXECKEY and DATALOCATION, but **never** set the LANGUAGE. CICS will do it for you according to information provided in the command level stub of a program.

Enhancements for TS Queues

There are three new functions in CICS TS 1.1.1:

- Up to now the name of a TS queue specified in QUEUE was restricted to a length of 8 bytes.

  Now in addition an up to 16 bytes long name can be specified along with the parameter QNAME.

- CEMT INQ TSQUEUE now shows much more information than before. It is like this:

```
Tsq(APPLNAME        ) Num(00036) Fle(00002304) Aux
   Max(00064) Min(00064) Xts(C1D7D7D3D5C1D4C5)
Tsq(IEOPADDQ        ) Num(00001) Fle(00000064) Mai
   Max(00064) Min(00064) Xts(C9C5D6D7C1C4C4D8)
```

  Please note the field 'Tsq' is shown 16 bytes long for QNAME content.

- If a 'b' is entered in column 1 in front of an entry, CEBR is called and shows the content of all records belonging to that queue.

CSD-File LSR Buffer Usage

During initial installation of z/VSE a CSD-File will be created automatically with a data and index CI size of 4K.

For CEDA functions at least 4 buffers of this size have to be available. If there are less then 4 buffers available some functions like

```
CEDA EXPAND GROUP(*)
```

will fail with CICS abend code 'AFCY'.

VSAM LSR Buffer Hashing - Much Improved Data In Memory

Starting with VSE/ESA 2.5 the new function of 'VSAM LSR Buffer Hashing' is available. This hashing for finding valid VSAM RBAs within an LSR subpool is a big performance enhancement. The CPU consumption for the search is independent of the number of LSR buffers in a subpool. So defining 1.000 or 10.000 (up to 32.768) buffers will introduce Data In Memory to a much higher degree.

This LSR Buffer Hashing does not depend on the CICS version but on VSAM only.

Automated Operation with CICS TS from Batch via CLISTs or REXX

The REPLID for a MSG xx is not fixed but varying based on internal CICS TS structure. This is independent of running in a static or dynamic partition. Therefore automated operation for performing CICS TS commands cannot rely on a fixed REPLID.

The following example shows how to send a command to a CICS TS using the partition Id:

```
msg x1,data='cemt perf shut'
```

The following example shows how to send a command to a CICS TS using the job name of a CICS TS:

```
msg DBDCCICS,data='cemt perf shut'
```

Of course it is possible to use a REXX procedure for getting the correct REPLID. The following example shows a CEMT I TASK:

```
ADDRESS CONSOLE
FC = GETMSG(MSG.,'MSG',,,5)  REXNORC'
'MSG C1'
FC = GETMSG(MSG.,'MSG',,,5)
FC = GETMSG(MSG.,'MSG',,,5)      <note - not an error - should be repeated!>
GETREPLYID(MSG.1) 'CEMT INQ TASK'
'DEACTIVATE CONSI'
EXIT
GETREPLYID:
ARG MESSAGE
POSITION = POS('-',WORD(MESSAGE,1))
IF POSITION > 0
THEN
REPLYID = SUBSTR(WORD(MESSAGE,1),POSITION+1)
ELSE
REPLYID=WORD(MESSAGE,2)
RETURN REPLYID
END
/+
```

This works for CICS TS running in any partition. For details about GetReplyId please refer to the 'VSE/REXX Reference' under 'Routing Messages From and Replies To a Specific Partition'.

Error Messages DFHEV1020S and DFHST0103 During CICS TS Start Up

If XRF is not active for CICS TS the following error message may be displayed at CICS TS startup:

```
DFHEV1020S  DFHCXRF Logical Unit SYS020 is either invalid for this
partition or is set to UA or IGN.
```

To avoid this error message just define

```
// ASSGN SYS020,SYSLST
```

If the DMF partition is not active the following error messages will be displayed:

```
DFHST0103 A0006CI1 A DMF error has occurred with return code X'10'
```

Either start up DMF partition or ignore this error message. Ignoring means not to have consistent statistics information for being processed by DFHSTUP for statistics and DFH$MOLS for monitoring.

GETVIS 24 Usage

Issuing the z/VSE AR command GETVIS 'xx' shows values for GETVIS 24 and ANY:

```
GETVIS USAGE    F2-24     F2-ANY                      F2-24     F2-ANY
 AREA SIZE:   11,260K    51,196K
 USED AREA:    8,696K    37,152K MAX. EVER USED:   11,260K    39,828K
 FREE AREA:    2,564K    14,044K LARGEST FREE:      2,528K    13,976K
```

As in the example shown, F2-24 AREA SIZE and F2-24 MAX. EVER USED have the same size. This is due to a test made by CICS TS during startup to check for the size of GETVIS 24 available to it. With this test the high water mark is set to its maximum value.

Therefore MAX. EVER USED is misleading in CICS TS systems. USED AREA is correct, it's the currently used size of GETVIS 24.

z/VSE console command:

```
GETVIS xx,RESET
```

resets the high water mark to the current allocated (USED) value. If issued directly after CICS TS startup, MAX EVER USED shows the correct value of the running system.

The above GETVIS xx,RESET command can also be issued by a program (e.g. in the PLTPI phase of CICS TS startup).

A sample of such a program is shown in ICCF Lib 59 (SKCICMD) and can be submitted for being assembled and cataloged. Program name is ARCCICS and transaction name has to be ARCM (described within this ICCF member). This program can be started within PLTPI and PLTSD and from a 3270 screen. In PLTPI state it RESETs the high watermark and in PLTSD state it gives just a GETVIS xx, to display the high watermark for this CICS TS.

Behavior of CICS TS in Case of an SOS Condition

There are new mechanisms to prevent CICS TS from going Short On Storage.

In case of SOS, CICS TS reduces the priority of newly attached tasks, to allow running tasks to finish faster. Beside of that, CICS TS will try to compress storage when reaching internally defined high water marks.

If a real stall condition occurs, the purging mechanism works as follows

There are two transaction parameters which control the possibility to purge tasks:

```
SPURGE(YES) now means SystemPURGEable - allow any kind of PURGE

DTIMOUT(mmss) defines the elapsed time after which PURGE can happen
```

Also in case of SOS, only tasks defined with **both** transaction parameters set correctly are eligible for being purged.
The former DFHSIT parameter ICVS has been removed.

Automated Start of DB2

It's an often used procedure to start up DB2 automatically during CICS start up by inserting the CIRB command into SYSIPT and by defining a sequential terminal in the DFHTCT:

```
DC    CL32'INPUT AND OUTPUT DTFS'
DFHTCT  TYPE=SDSCI,                                          *
      DEVADDR=SYSRDR,                                        *
      DEVICE=2540,                                           *
      DSCNAME=READER
DFHTCT  TYPE=SDSCI,                                          *
      DEVADDR=SYSLST,                                        *
      DEVICE=1403,                                           *
      DSCNAME=PRINTER
DC    CL32'S A M A  LINE AND TERMINAL ENTRY'
DFHTCT  TYPE=LINE,                                           *
      ACCMETH=SEQUENTIAL,                                    *
      TRMTYPE=CRLP,                                          *
      ISADSCN=READER,                                        *
      OSADSCN=PRINTER,                                       *
      INAREAL=88
DFHTCT  TYPE=TERMINAL,                                       *
      TRMIDNT=SAMA,                                          *
      TRMPRTY=32,                                            *
      TRMSTAT=(TRANSCEIVE)
```

With CICS TS an additional /*-line has to be inserted prior to user specified transactions, e.g. CIRB:

```
// EXEC DFHSIP,SIZE=DFHSIP,PARM='SIT=C3,START=COLD,SEC=NO,STATRCD=OFF,S*
               VA=NO,NEWSIT=YES,DSALIM=8M,EDSALIM=30M,SI',DSPACE=2M,OS3*
               90
/*        <------ this line has to be inserted
CIRB PASSWORD,8,XXXO3,1,GER,PRODDBI
/*
```

If this /* is not inserted, no error message will be sent to the console. The SYSIPT data for CIRB or other user transactions will be ignored completely.

Translating and Compiling EXCI-programs

A CICS TS controlled batch to CICS TS communication was introduced via the EXCI function. For translating and compiling such batch programs containing EXEC CICS commands, the normal translator/compile jobs for CICS programs available in ICCF library 2 can be used with minor changes. These changes are:

- the name for Assembler is SKEXCIAS.

- the name for C/VSE is SKEXCICN.
- the name for COBOL/VSE is SKEXCICV.
- the name for PLI/VSE is SKEXCIPV.

Translating Programs Containing System Programming Commands

The new translator option 'SP' must be specified for application programs that contain special (System Programming) CICS commands or they will be rejected at translate time. These commands are ACQUIRE, COLLECT, DISABLE, DISCARD, ENABLE, EXTRACT, INQUIRE, PERFORM, RESYNC and SET.

Sample Application Programs

Information about CICS supplied sample application programs is now collected in the new manual 'Sample Application Guide', SC33-1713.

In short, sample program names start with

```
DFH$A for Assembler,
DFH0C for COBOL,
DFH$D for C,
DFH$P for PL/I.
```

These programs are available as source in library PRD1.BASE.

Hints about Statistics

DFH0STAT - Statistics Program for Test Purposes

DFH0STAT.C and the related map DFH0STM.A can be found in PRD1.BASE. First process DFH0STM as map and after that DFH0STAT as COBOL program. Prerequisite for compiling DFH0STAT is the 'COBOL FOR VSE' compiler. To have DFH0STAT available to customers not having this compiler, a link job named DFH0STAT in library 59 can be used.

This program collects all statistics information with EXEC CICS commands:

```
EXEC CICS COLLECT STATISTICS
```

and writes it via EXEC CICS SPOOLWRITE to VSE/POWER List Queue (name is that of the CICS-Startup-Job).

Its usage is best for test purposes.

It should not be used for a production CICS because, for example:

- there are no statistics information available for previously closed datasets, for autoinstall terminals, which made a logoff etc.
- all statistics counters are reset at statistics interval occurrence and at midnight or with a CEMT PERFORM STATISTICS ALL RESETNOW . It is possible to change the statistics interval with CEMT SET STATISTICS INTERVAL(230000) to avoid the reset of counters during the day. This can be accomplished by a program, which executes an

```
EXEC CICS SET STATISTICS INTERVALHRS(23)
```

DFHSTUP - Statistics Program for Production Purposes

Statistics data collected with DMF are correct in any case. Therefore production statistics should be printed with DFHSTUP only. To limit the statistics output, use parameter SUMMARY in normal cases. A skeleton for such a print job (SKDMFPR) is in ICCF Library 59. To start DMF in a z/VSE partition use skeleton SKDMFST in Lib 59.

Statistics Information Taken from a System Dump

You can select some statistics information from a CICS driven system dump. This is accomplished by the INFOANA program.

The following example can be used as a skeleton for own customizing. It was generated with Interactive Interface 'STORAGE DUMP MANAGEMENT' (option 4.3) and 'ANALYZE CICS DUMP' (option 9).
Please refer to 'CICS TS Problem Determination Guide' for a detailed description of the level (0,1,2 or 3), that can be specified for each Domain.

To get SUMMARY information, use dump level 1, but not all Domains allow this summary level. To suppress information use dump level 0. Dump level 2 means detailed information, dump level 3 means SUMMARY and DETAIL.
E.g. PG=1 will produce statistical information collected by the Program Domain.

```
// JOB DMPACD22  ANALYZE CICS/TS DUMP
// EXEC PROC=DTRINFOA
// EXEC INFOANA,SIZE=INFOANA,OS390
  SELECT DUMP MANAGEMENT
    DUMP NAME SYSDUMP.F2.DF200019
        RETURN
  SELECT DUMP VIEWING
        CALL DFHPD430 DATA AP=0,KE=0,DS=0,TR=0,LD=0, -
    DATA AI=0,BR=0,CC=0,CP=0,CSA=0,DD=0,DH=0, -
    DATA DM=0,DU=0,FCP=0,ICP=0,IE=0,IND=0,  -
    DATA JCP=0,LM=0,ME=0,MN=0,MRO=0,PA=0,      -
    DATA PCT=0,PG=1,PR=0,RM=0,SM=0,SO=0, -
    DATA SSA=0,ST=0,SZ=0,TCP=0,TDP=0,TI=0, -
    DATA TMP=0,TSP=0,UEH=0,US=0,WB=0,XM=0,    -
    DATA XRF=0,XS=0
        RETURN
    DUMP NAME SYSDUMP.F2.DF200019
  SELECT DUMP VIEWING
        PRINT FORMAT
        RETURN
  SELECT END
/*
/&
```

Figure 108. Sample job for INFOANA

Shutdown Process Using DFH$SDAP

In ICCF library 59 the a skeleton DFH$SDAP establishes shutdown assist program DFH$SDAP. This program allows a staged shutdown in four steps, active transactions are

- Forced
- Forcepurged
- VTAM is forceclosed
- and finally SHUTDOWN IMMEDIATE is issued.

The time delay between the different steps can be changes in the source code DFH$SDAP.A in PRD1.BASE. Please refer to the skeleton for details on how to define the program and setup the shutdown PLT.

Some Hints about Tracing

The Trace Domain in CICS TS is completely new. From a debugging point of view the new auxiliary trace is much better than the old one. Some examples for this:

- Exception conditions (abends etc.) are marked with an '*EXC*'. So it is a good practice to search for '*EXC*.
- If a program is translated with the translator option DEBUG, the EI trace level 2 trace shows the line number of the source program. Of course this DEBUG option should be used during test state only - due to DEBUG the program becomes much larger in size and needs more CPU power. EI trace level 2 can be set via CETR and PF4 - 'components'.
- Normal trace level 1 does not show the resource name for an EXEC CICS command. To see the resource name, use EI trace level 2. Now the resource name and even some data are shown. Example:

```
ENTRY READ                 'FILEA   ' ........................
ENTRY READ_INTO            02122120 , ........................
EXIT  READ_INTO/OK         50,50,00000000,,LENGTH_OK
EXIT  READ                 'FILEA   ' AT X'0063B1C8',' 000100W. DAVIS'
```

  FILEA is the file name and ' 000100W. DAVIS' is the record content.

New Dump Tables and Dump Hints

These new tables control whether a dump has to be taken, if a CICS shutdown is necessary and how many dumps for a specific dump code should be written.

You should consider some points:

- In case of an ASRA or ASRB a transaction **and** a system dump is taken by default.
  To disable the system dump for ASRA and ASRB, the following two entries have to be applied to the system dump table, either via CEMT or EXEC CICS.

```
CEMT SET SYDumpcode(AP0001) ADD NOSYSDUMP
CEMT SET SYDumpcode(SR0001) ADD NOSYSDUMP
  or
EXEC CICS SET SYSDUMPCODE(AP0001) ADD NOSYSDUMP
EXEC CICS SET SYSDUMPCODE(SR0001) ADD NOSYSDUMP
```

- In case of a storage violation CICS will not shutdown by default. If STGRCVY(NO) is in effect, CICS abends the transaction but does not release the violated storage. If STGRCVY(YES) is set, CICS tries to recover the violated 'check zone' and continues to run the task.

  If a shutdown of CICS TS is needed in these cases, the following entries have to be put to the system dump table:

```
CEMT SET SYDumpcode(SM0102) ADD SHUTDOWN
CEMT SET SYDumpcode(SM0103) ADD SHUTDOWN
  or
EXEC CICS SET SYSDUMPCODE(SM0102) ADD SHUTDOWN
EXEC CICS SET SYSDUMPCODE(SM0103) ADD SHUTDOWN
```

  After a cold start these entries have to be established again.

A sample program showing these commands is provided in ICCF lib 59, named SKSUPDMP.

- By specifying the SIT parameters SYDUMAX=1 and TRDUMAX=1 the limit of number of dumps per system and transaction dump code is set to one. Especially system dumps should be limited in this way. Default is 999.
- Program checks are shown as '0Cx'

  Does a program check occur in a CICS TS Domain other than the Application Domain, the CICS TS abend code is AKEA: The Kernel Domain is setting the abend exits and returns control to the Domain with the failing instruction. AKEA is set, preceded by 0Cx.

  control from Kernel Domain and calls DFHSRP which sets the ASRA abend code.

With VSE/ESA 2.7 there is a new REXX procedure named REXDFHDU, which allows you to print transaction dumps selectively.

With initial install of VSE/ESA 2.7 two jobs (PRTDUMPA and PRTDUMPB) are being loaded into the POWER reader queue. These jobs are also stored in IJSYSRS.SYSLIB as PRTDUMPA.Z and PRTDUMPB.Z.
In the PARM statement of these job the 'CICS=CICSICCCF' has to be adapted to the individual 'CICS=jobname'.
For a second CICS TS the DLBL cards must be adapted to the correct file id of the dump datasets.

CICS catalogs DFHGCD and DFHLCD

There are two new datasets used by CICS TS for its own purposes:

- DFHGCD is the Global CICS Dataset. It contains information about installed definitions and warm/emergency start information. It is shared with any XRF tracking partition, but CANNOT be shared between normal CICS partitions.

  The DFHRSD is only used for backout information during emergency restart.
- DFHLCD is the so called Local CICS Dataset and contains records that describe the status of the Domains and dump options. Each CICS TS partition (normal or XRF) has to have an own DFHLCD.

The Global and Local catalogs should always be defined and initialized together to avoid CICS startup errors.

Defining and initializing the Global and Local catalogs will force a COLD start also, if START=AUTO is used. This cold start is also known as 'ice cold start'.

Skeletons for delete/define both catalogs are in ICCF lib 59, named SKGCDFIL and SKLCDFIL.

SVA Usage

There are three types of SVA eligible phases:

- Mandatory CICS TS phases

```
DFHCSCV   Always required - CICS TS SVC
DFHDSPEX  Always required - Dispatcher POST exit
DFHCSEOT  Always required - EOJ cleanup

DFHCDDAN  Required for XRF

DFHIRP    Required for MRO (IRC program)
DFHIRW10  Required for MRO (work exit)
DFHSCTE   Required for MRO (SCTE control block)

DFHDTSVC  Required for Data Tables (SVC)
DFHDTSAN  Required for Data Tables (Anchor control block)
```

  A failure to load these phases into the SVA by SET SDL will cause CICS TS initialization failures.

  – LIST=$SVACICS from PRD1.BASE must be used.

- Non-Mandatory CICS TS phases

  Loading of these phases is done by SET SDL also.

  Usage of SVA resident phases is controlled by SIT-parameter SVA. Since VSE/ESA 2.6 the DFHSITSP in lib 59 specifies SVA=YES and contains in the CICS-startup skeleton SKCICS the PRVMOD member

```
$$$$ SLI MEM=DFH$SVEX.J,S=PRD1.BASE
```

  as overwrite statement.
  If SVA=YES is specified but a requested SVA-phase is not loaded into SVA, it will be loaded into CICS TS (E)DSAs and message DFHLDLD1 or DFHLDDMI is issued. Specify SVA=YES and startup CICS. The DFH-messages tell you which phases should be really loaded into the SVA.

- Non-Mandatory Non-CICS TS phases (e.g. LE-phases)

  To use such SVA resident phases specify SIT parameter SVA=YES plus USESVACOPY(YES) in each RDO program entry.
  Be aware of the following: program products may supply RDO definitions with USESVACOPY(NO) as default.
  CICS will load a private copy of the phase even if it exists in the SVA. Update the eligible RDO program definitions.
  Starting with VSE/ESA 2.6 some LE/VSE phases will be loaded into SVA as default. If not needed, please adapt the SVA load list.

Security Hints

- If a user cannot logon or isn't logged on, the default userid will be assigned to him. The name of this default userid is set in DFHSIT with DFLTUSER=username. In DFHSIT this defaults to CICSUSER, which is defined in IESCNTL and which has FULL access rights (1-64).
  It is therefore strongly recommended in a production environment, to create another default user equivalent to CICSUSER, and change the access rights according to individual needs and change DFLTUSER in DFHSIT.
- Default security of CNSL is CICSUSER. To have special security for CNSL, it is recommended to define a special user (e.g.named CNSL) to BSM or ESM with its own security settings and to alter CNSL terminal definition to USERID=CNSL. With this CNSL gets the security settings of user CNSL.
- If Interactive Interface users need to have full console support (e.g. use of CEMT-Commands for CICS TS partitions) they have to have a dedicated console definition. Actions needed:

  Copy CNSL Terminal definition in RDO-Group VSESPG to an own group with

  COPY TO(IIKONS) AS(user). 'user' is the II userid.

  This new TERMINAL(user) needs to be modified with:

  NETNAME(user),CONSNAME(user).

  After installing this new definition all console functions are available to this user.
- Transaction security can be dedicated to a CICS TS region for BSM users (II 2.8 dialogue on z/VSE 3.1.1, this is 285 in case security is not migrated to the BSM based security concept, 2811 if migrated). If using the REGION option, please note:
  Set SECPRFX=YES in DFHSIT (NO is default)
  The REGION name is equivalent to the USER=username in // ID card of CICS TS startup.

Migrating TRANSIDs With TCLASS From CICS 2.3 to CICS TS

With CICS 2.3 number of attached tasks per class is controlled by CMXT values in DFHSIT. Under CICS TS CMXT is obsolete. These numbers are now controlled by TRANCLASS definitions (group DFHTCL).

Be aware to adapt the old CMXT values to the new TRANCLASSes DFHTCL01 to DFHTCL10. This can be done by copying these TRANCLASS definitions from group DFHTCL to a new user group. These copied definitions has to be altered according to CMXT values and added to the group list, defined in DFHSIT as GRPLIST. If one forget to adapt DFHTCLxx definitions (which default to Maxactive: 001) and this maxvalue is reached, no warning will be displayed. The auxtrace shows in this case:

XM 0B02 XMQC EXIT TCLASS_ACQUIRE/EXEPTION QUEUED

Information About LE/VSE 1.4.3 and higher

- Recommended CICS-Wide Run-Time Options

  A setting of value [n x [4096 minus 16]] bytes instead of [n times 4096] bytes for the second LE Run-Time Options such as ANYHEAP, HEAP and STACK, addressing can provide the following benefit:

  Size [n times [4096 minus 16]] bytes gives additional space required by the CICS/VSE storage accounting areas (SAAs) to fit into one page frame of 4K. The same applies for CICS TS (where the corresponding areas to SAA are termed "crumbled zones"). The SAAs and crumbled zones are used to detect storage violations.
  This minimizes the storage fragmentation. The general rule is 4k or a multiple of 4K minus 16 bytes.

- Activating Changed CICS-Wide Run-Time Options

  By executing the new transid **NEWC** you can update new cataloged LE/VSE runtime options for CICS. This is done while the CICS system is still processing online transactions.
  This transaction is available with CICS TS and CICS/VSE 2.3.

- Printing CICS-Wide Run-Time Options to Console

  This function allows you to print LE/VSE CICS-wide run-time options to your z/VSE console.
  To show the run-time options issue tranaction **ROPC** which will invoke program EDCYCROP. This transaction is available with CICS TS and CICS/VSE 2.3.

- Change CICS-Wide Run-Time Options dynamically with transaction CLER

  With LE/VSE 1.4.3 there is a new transid **CLER** provided. This transaction allows the system programmer not only to display the current LE/VSE runtime options on a 3270 screen but also to alter them dynamically.
  To alter STORAGE values is not allowed because of LE internal reasons. Altered values will be active immediately but are valid for this CICS system only. In addition these changes are temporarily only and will disappear after CICS is being recycled.
  This transaction is available with CICS TS and CICS/VSE 2.3. Of course the CSD-Group CEE should be recreated for CICS/VSE 2.3 if shared CSD file is not used.

- To switch off exit IESPDATX, compile CEECXTAN.A using CEESCEXT in ICCF library 62 after you removed or commented out following line:

  ```
  CEEXART  TERMXIT=IESPDATX,TYPE=POSTDUMP
  ```

  Make sure after you compiled, that the phase is in the search chain ahead of PRD2.SCEEBASE for the related CICS partition.

  Note: Disable IESPDATX exit for LE OLPD in case IUI is not used. Deactivation avoids invocation of LE abend processing.

CICS Applications Using TCP/IP Connections

If a CICS application uses any TCP/IP connection you must add the statement:

```
// OPTION SYSPARM='nn'
```

to the JCL for these jobs. The value 'nn' (the system ID) is contained in the:

```
// EXEC IPNET,SIZE=IPNET,PARM='ID=nn,INIT=... '
```

statement of your TCP/IP startup job.

Note: The value of '00' is the default for the system ID. If you accept this default, you are not required to add the statement
// OPTION SYSPARM='00'.

Examples for such applications are:

- CICS Web Support
- CICS Transaction Gateway over TCP/IP
- MQSeries for VSE/ESA over TCP/IP
- CICS programs using socket interface

Telnet Definitions for Terminals With Extended Data Stream (EXTDS)

If a CICS application needs extended data stream functions like, for example, color, highlighting, or reverse video, then the following definitions have to be made. Note: file transfer with IND$FILE needs EXTDS also.

The correct model for needed EXTDS functions has to be defined in the APPL definition and in the TELNETD definition. An example for these definitions looks like the following:

```
TNICCF01 APPL  AUTH=(ACQ),DLOGMOD=SP3272EN,MODETAB=IESINCLM

DEFINE
TELNETD,ID=ICCF,TARGET=A0006CI1,TERMNAME=TNICCF,PORT=6023, -
      LOGMODE=SP3272EN,COUNT=4,POOL=YES
```

CICS TS autoinstall program IESZATDX selects the corresponding TYPETERM for the terminal definition.

In case that these LOGMODE definitions are missing, TCP/IP will use a default LOGMODE with no extended data stream.

Hints for CICS Transaction Gateway with TCP/IP Connection

In addition to LU 6.2 connection, with VSE/ESA 2.6 connection via TCP/IP is available. Please notice, that there is a new transaction CIEP needed for this support. So this transaction has to be known to the External Security Manager (BSM or others).

Where to Find Corrective Service for CICS TS

An often raised question is: where can I find preventive and corrective service for CICS TS in the Internet ? The answer is to go to the URL shown below:

http://www14.software.ibm.com/webapp/set2/psearch/search?domain=sysz

and type in an error message or indication or the product number 5655VSE00 of CICS TS to get all related APARs (and PTFs of course).
After finding a valid APAR it is possible to download the related PTF(s) including PRE- and CO-requisites via file transfer. The procedure for download is described in a note sent to the E-Mail address specified with the userid.

Chapter 16. Crypto and SSL

Cryptographic support

Information about cryptographic support in z/VSE can be found in the VSE Administration book, chapters

1. Implementing Hardware Cryptographic Support
2. Implementing Hardware-Based Tape Encryption
3. Implementing the Encryption Facility for z/VSE

Further information is provided on the VSE Homepage in the Security section:

www.ibm.com/systems/z/os/zvse/documentation/security.html

1. How to setup SSL with CICS Web Support (PDF, 1.4MB)
2. How to setup Secure Telnet with VSE (PDF, 1.7MB)
3. How to setup Secure FTP with VSE (PDF, 1.2MB)
4. How to setup cryptographic hardware for VSE (PDF, 1.1MB)

These documents are written in Redbook style and provide much hints and tips like information.

Much information about Security, SSL/TLS, and cryptography is also provided in these two IBM Redbooks:

1. Security on IBM z/VSE, SG24-7691

   http://www.redbooks.ibm.com/abstracts/sg247691.html?Open

2. Enhanced Networking on IBM z/VSE, SG24-8091

   http://www.redbooks.ibm.com/abstracts/sg248091.html?Open

Display crypto environment

Information about the current crypto environment is displayed on the console via the STATUS=CR command. Example:

```
msg fb,data=status=cr
AR 0015 1I40I  READY
FB 0011 BST223I CURRENT STATUS OF THE SECURITY TRANSACTION SERVER:
FB 0011 CRYPTO DEVICE DRIVER STATUS:
FB 0011   AP CRYPTO SUBTASK STARTED ......... : YES
FB 0011   MAX REQUEST QUEUE SIZE ............ : 5
FB 0011   MAX PENDING QUEUE SIZE ............ : 3
FB 0011   TOTAL NO. OF AP REQUESTS .......... : 12345
FB 0011   NO. OF POSTED CALLERS ............. : 12345
FB 0011   AP-QUEUE INTERRUPTS AVAILABLE ..... : YES
FB 0011   AP-QUEUE INTERRUPTS STATUS ......... : DISABLED
FB 0011   AP CRYPTO POLLING TIME (1/300 SEC).. : 1
FB 0011   AP CRYPTO WAIT ON BUSY (1/300 SEC).. : 75
FB 0011   AP CRYPTO RETRY COUNT .............. : 5
FB 0011   AP CRYPTO TRACE LEVEL .............. : 3
FB 0011   TOTAL NO. OF WAITS ON BUSY ......... : 0
FB 0011   CURRENT REQUEST QUEUE SIZE ......... : 0
FB 0011   CURRENT PENDING QUEUE SIZE ......... : 0
FB 0011   ASSIGNED APS : PCICC / PCICA ....... : 0 / 0
FB 0011                  CEX2C / CEX2A ....... : 0 / 0
FB 0011                  CEX3C / CEX3A ....... : 0 / 0
FB 0011                  CEX4C / CEX4A / CEX4P : 1 / 1 / 0
FB 0011                  PCIXCC .............. : 0
FB 0011     AP  1 : CEX4A   - ONLINE
FB 0011     AP  2 : CEX4C   - ONLINE
FB 0011   ASSIGNED AP QUEUE (CRYPTO DOMAIN)... : 9
FB 0011   NO. OF AVAILABLE CRYPTO DOMAINS .... : 16
FB 0011 END OF CRYPTO DEVICE DRIVER STATUS
```

The STATUS=CPACF command displays the CPACF-related status.

```
msg fb,data=status=cpacf
AR 0015 1I40I  READY
FB 0011 BST223I CURRENT STATUS OF THE SECURITY TRANSACTION SERVER:
FB 0011 CPU CRYPTOGRAPHIC ASSIST FEATURE:
FB 0011   CPACF AVAILABLE ................... : YES
FB 0011   INSTALLED CPACF FUNCTIONS:
FB 0011     DES, TDES-128, TDES-192
FB 0011     AES-128, AES-192, AES-256, PRNG
FB 0011     SHA-1, SHA-256, SHA-512
FB 0011     KMAC_DES, KMAC_TDES128, KMAC_TDES192
FB 0011     CMAC_DES, CMAC_TDES128, CMAC_TDES192
FB 0011     XTS_AES_128, XTS_AES_256
FB 0011   PROTECTED KEY CPACF FUNCTIONS:
FB 0011     ENCR_DES, ENCR_TDES128, ENCR_TDES192
FB 0011     ENCR_AES128, ENCR_AES192, ENCR_AES256
FB 0011     KMAC_ENCR_DES, KMAC_ENCR_TDES128, KMAC_ENCR_TDES192
FB 0011     CMAC_ENCR_DES, CMAC_ENCR_TDES128, CMAC_ENCR_TDES192
FB 0011     XTS_ENCR_AES128, XTS_ENCR_AES256
FB 0011   ENCRYPTION MODES:
FB 0011     ECB, CBC, CFB, OFB, CTR
FB 0011 END OF CPACF STATUS
```

Tracing

Hardware crypto support is given either

1. via crypto cards (PCICA, PCIXCC, Crypto Express) providing asymmetric cryptographic functions (RSA encrypt and decrypt), or
2. via the CPU Cryptographic Assist feature (CPACF) providing symmetric cryptographic functions, like DES, Triple-DES, AES, and hash functions, like SHA-1.

Availability of particular functions depend on the System z processor.

CPACF functionality is given by a set of machine instructions described in the Principles of Operation book and can therefore be used by any vendor program directly. Tracing is only possible on Firmware level.

Crypto cards are accessed via a vendor API implemented in phase IJBHCDRV. Tracing is possible via the APTRACE operator command.

TCP/IP for VSE/ESA 1.5E or later transparently performs cryptographic functions in hardware when available.

Tracing of traffic to crypto cards is controlled via the APTRACE command Example:

```
msg fb,data=aptrace=0
AR 0015 1I40I  READY
FB 0011 1J034I CRYPTO TRACE LEVEL SET TO 0.
```

There are three trace levels:

- 0 - full trace (errors, warnings, and informational messages)
- 1 - warning and error messages
- 2 - error messages only
- 3 - trace off (default)

When now for example connecting to VSE via SSL, trace is written to the console. Example:

```
R1 0045 -> IJBHCDRV, fcode = 2
R1 0045 <- IJBHCDRV, rc = 0
FB 0095 IJBCRYPT: func_code = FUNC_CRYPT            <- crypto request
FB 0095 -> Crypt ...
FB 0095 -> EnqToCD, psmid = 0578D960
FB 0095 -> SelDev, type_code = 5001D000
FB 0095 <- SelDev, device_nr = 0                    <- AP 0 is used
FB 0095 -> EnqToAP, dev_nr = 0                      <- enqueuing request
FB 0095  q_num  = 00000004                          <- AP queue is 4
FB 0095 <- EnqToAP, dev stat = 00000001 = ONLINE    <- AP 0 is online
FB 0095 dev_caller_count = 1
FB 0095 <- EnqToCD, rc = 0                          <- successfully enqueued
FB 0095 -> DeqNext() ...
FB 0095 No. of callers on AP 0 (CEX2A  ): 1
FB 0095 -> DeqFrCD,   index = 0                     <- dequeing response
FB 0095 <- Deq(), ccode = 0            AP status = 80000000
FB 0095 <- DeqFrCD, rc = 0
FB 0095 -> ProcResp, psmid = (0578D960)
FB 0095 -> ConvResp,  pRsp = 0586C800
FB 0095 Response header(8/16 bytes): ----------------
FB 0095 0586C800: 00800090 00000000 00000000 00000000
FB 0095 ----------------------------------------------
FB 0095 Got a good CEX2A response.                  <- response was good
FB 0095 reply_code = 00000000
FB 0095 response code was mapped to rc = 1
FB 0095 <- ConvResp, rc = 0
FB 0095 -> PostCaller, pRequest = 0578D950          <- post TCP/IP partition
FB 0095 No. of callers on AP 1 (CEX2C  ): 0         <- no more callers for AP 0
R1 0045 IESC1023I CLIENT CONNECTED FROM IP:   9.152.216.53
FB 0095 Crypto task waiting for ECB posting at 0586DFC8
R1 0045 IESC1028I CLIENT SESSION ESTABLISHED FOR USER: JSCH
```

Disabling the use of crypto cards

Using cryptographic hardware can be disabled via a TCP/IP $SOCKOPT phase. Example:

```
* $$ JOB JNM=$SOCKOPT,CLASS=0,DISP=D
* $$ LST CLASS=A
// JOB $SOCKOPT
// OPTION CATAL
// LIBDEF *,SEARCH=PRD2.TCPIPC
// LIBDEF *,CATALOG=PRD1.BASE
// EXEC ASMA90,SIZE=ASMA90,PARM='SZ(MAX-200K,ABOVE)'
         PUNCH    ' PHASE $SOCKOPT,* '
$SOCKOPT CSECT
         SOCKOPT  CSECT,                   Generate a csect           X
               SSLFLG2=$OPTSNHC,           SSL do not use hw-crypto   X
         END   $SOCKOPT
/*
// EXEC LNKEDT,SIZE=512K
/*
/&
* $$ EOJ
```

Note that RSA keys with length 2048 bit or higher require the use of crypto cards.

Disabling the use of CPACF

Following JCL disables the use of the CPACF feature.

```
* $$ JOB JNM=SOCKOPT1,CLASS=0,DISP=D
// JOB SOCKOPT
*
* * CREATE A $SOCKOPT.PHASE THAT SUPPRESSES THE
* * USE OF THE KMC INSTRUCTION
*
// OPTION CATAL
// LIBDEF *,SEARCH=PRD2.TCPIPC
// LIBDEF *,CATALOG=PRD1.BASE
// EXEC ASMA90,SIZE=ASMA90,PARM='SZ(MAX-200K,ABOVE)'
         PUNCH    ' PHASE $SOCKOPT,* '
$SOCKOPT CSECT
         SOCKOPT  CSECT,SSLFLG2=$OPTSNZA
         END   $SOCKOPT
/*
// EXEC LNKEDT,SIZE=512K
/*
/&
* $$ EOJ
```

Note that some symmetric algorithms, like SHA-224/256/384/512 require the use of CPACF.

Forcing the use of CPACF

Following JCL forces the use of the CPACF feature.

```
* $$ JOB JNM=SOCKOPT2,CLASS=0,DISP=D
// JOB SOCKOPT2
*
* * THIS JOB WILL CREATE A $SOCKOPT.PHASE THAT FORCES THE
* * USE OF THE KMC INSTRUCTION
*
// OPTION CATAL
// LIBDEF *,SEARCH=PRD1.BASE
// LIBDEF *,CATALOG=PRD1.BASE
// EXEC ASMA90,SIZE=ASMA90,PARM='SZ(MAX-200K,ABOVE)'
         PUNCH    ' PHASE $SOCKOPT,* '
$SOCKOPT CSECT
         SOCKOPT  CSECT,SSLFLG2=$OPTSFZA
         END   $SOCKOPT
/*
// EXEC LNKEDT,SIZE=512K
/*
/&
* $$ EOJ
```

Chapter 17. IWS File Transfer

Basic Problem Solving Questions

1. Is the PC configured as CUT (Control Unit Terminal) or DFT (Distributed Function Terminal)? Transaction INWQ displays CUT or DFT.
   For each device associated with a host session the Extended Data Stream (EXTDS) feature must be specified. This is the case for device 24x80Q, TYPETERM VSE3278Q.
   You can define a terminal in DFT mode using dialog *Configure Hardware* (fastpath 241) and adding for example a *Local non SNA Terminal* as device *24x80Q*.
2. How is the PC attached to the host (ICA, DPA, 3174, 3274-31A, Telnet-TCP/IP, HTERM)?

   If the PC is attached by a DPA only CUT mode is supported.
3. For Telnet, the default terminal logmodes are without extended data stream, thus DFT is not possible. It is recommended to define the Telnet daemon with query-able terminals:

   ```
   LOGMODE3=SP3272QN,LOGMODE4=SP3272QN,LOGMODE5=SP3272QN
   ```

   Or with explicitly defined logmodes with extended data stream set on:

   ```
   DEFINE TEL,ID=MYTEL,TAR=DBDCCICS,TERM=T1000,CO=20,LOGMODE=SP3272EN, -
          LOGMODE3=SP3273EN,LOGMODE4=NSX32704,LOGMODE5=NSX32705
   ```

   Note: For Telnet, each terminal model needs to be defined, for a model 3 type, LOGMODE3 should be specified.
4. What is the model of the PC?
5. What 3270 emulation program is running on the PC ?
6. What release of VSE is running ? LIBRARIAN may be used to list SPLEVEL.PROC in IJSYSRS.SYSLIB.

```
568606601 65C  VSE/SP Unique VSE ESA 2.6.X
568606601 75C  VSE/SP Unique VSE ESA 2.7.X
5686CF801 81C  z/VSE SP Unique 3.1.X
5686CF801 91C  z/VSE SP Unique 4.1.X
5686CF801 01C  z/VSE SP Unique 4.2.X
5686CF801 02C  z/VSE SP Unique 4.3.X
5686CF901 51C  z/VSE SP Unique 5.1.X
5686CF901 52C  z/VSE SP Unique 5.2.X
5686VS601 61C  z/VSE SP Unique 6.1.X
5686VS601 62C  z/VSE SP Unique 6.2.X
```

Figure 109. Current z/VSE Releases

   VSE file transfer is not called IND$FILE however it does contain a transaction called IND$, or in case of DBCS APVU. The VSE file transfer phase is called INWPCCOM. Some user refer to the VSE file transfer function as IWS (Intelligent Workstation Support), or IUI file transfer or CICS file transfer.
7. What messages are received on the PC ?

8. What messages are received upon toggling back to the HOST session ? There may also be messages in the INSPECT MESSAGE LOG dialog.
9. Do both the SEND and RECEIVE fail ?
10. What is the exact SEND or RECEIVE command that was entered including any options ?

```
SEND PC.FIL FILENAME (FILE=HTF ASCII CRLF REPLACE
RECEIVE PC.FIL (FILE=LST KEEP
```

Figure 110. Send/Receive Examples

An easy way to find out is the so called IWS trace, see below.

11. Where is the send or receive directed to or from ?

    File transfer can be performed to or from:

    - VSE Host Transfer File (HTF)

      This file is a large VSAM file that contains separate entries for each file that a user sends to it. The entries are connected to a directory within the HTF for that user. VSE provides dialogs to move the files from this large HTF to other VSAM files or ICCF members.

    - CICS Temporary Storage

      A CICS application can then manipulate the TS queue.

    - VSE/POWER queues
      - A file can be sent or received from the POWER LIST queue (output from HOST jobs).
      - A file containing a VSE job can be sent to the POWER RDR queue.
      - A file can be received from the POWER PUNCH queue.

    - VSE library

Documentation Needed to Pursue Failures

One of the following traces will be required to determine if the error is being caused by the VSE software or the 3270 emulator software. For abends in VSE code a CICS AUX trace may also be required.

1. IWS TRACE:

   This is available as an option on the SEND/RECEIVE command. It is documented in the VSE Guide For Solving Problems.

2. VTAM BUFFER TRACE:

   The VTAM buffer trace can be started and stopped by entering the following commands at the VSE system console:

```
F NET,TRACE,TYPE=BUF,ID=TERMID
F NET,NOTRACE,TYPE=BUF,ID=TERMID
```

Figure 111. VTAM Commands to Get VTAM Buffer Trace

For TERMID use the entire VTAM NETNAME. Use TPRINT to print the trace. The following messages will be issued:

- IST907A - SNAPSHOT MODE TPRINT ENTER Y OR N

  Reply 'NO' to this message.

- IST905A - ENTER TRACE PRINT OPTIONS OR 'CANCEL'

  Reply 'PRINT BUF=ALL'

3. SDAIDS SIO/IO TRACE WITH CCWD:

   This job will set up the SDAIDS trace:

```
// JOB SDAID
// EXEC SDAID
OUTDEV T=tcuu
TRACE SIO AR=Fn UN=cuu OUTP=(TOD CCWD=2000)
TRACE IO AR=Fn UN=cuu OUTP=(TOD CCWD=2000)
/*
/&
```

Figure 112. Get an SDAID Trace Job Sample

   Fn is the CICS/ICCF partition. *tcuu=tape* device for trace output. *UN=cuu* is the line, terminal, or controller being traced. If controller is being traced be sure to reduce the traffic.

   a. Submit the above job.
   b. Enter STRTSD.
   c. Recreate the error.
   d. Enter STOPSD then ENDSD.
   e. Use DOSVSDMP to print the trace.

4. TCP/IP File Transfer Program

   For more information on how to trace TCP/IP FTP see the manual "TCP/IP for VSE/ESA - IBM Program Setup and Supplementary Information and refer to the topics TRACES, DEFINE TRACE command, and for output DEFINE LINK.

   For security consideration, refer also to TCP/IP security program BSSTIDX. This will avoid the need to specify each user performing FTP in the TCP/IP startup deck IPINITxx. The exit program is defined as follows:

```
DEFINE SECURITY,DRIVER=BSSTISX
```

Figure 113. Activate BSSTIDX security exit for TCP/IP

   In TCP/IP, SEC must be set ON.

Usage or Configuration Errors, Emulator Problems

1. MSGINW0002I from a module name ending with '0' and the PC is connected in DFT Mode.

   This is a generation and tailoring error, the module names ending in '0' are used in CUT (Control Unit Terminal) Mode. The user has to make sure that the PC, controller, VTAM tables, and the CICS terminal entries are configured for EXTENDED DATA STREAMING when using the PC as a DFT (Distributed Function Terminal).

2. MSGINW0002I with RC=0001 and the error description for the affected module is 'INVALID REPLY (NO AID ENTER)'.

   This means the response buffer did not begin with X'7D' (7D is the 3270 AID byte for an Enter Key). This message will also be issued if the user tries to toggle back to the host or if he uses Ctrl-Break to break a hang before file transfer is complete. The Ctrl-Break will send in data that will not begin with x'7D'. The reason for the hang should be investigated.

3. File transfer AID BIT problems: MSGINW0002I TRANS10 HANGS

   MSGINW0002I TRANS10 hangs can occur if the 3174 Control Unit Hardware configuration Question #125 bit 6, 'FILE TRANSFER AID', is not set on. Without this bit turned on a RECEIVE will work but SEND will fail. A VTAM Buffer Trace will show empty data buffers.

   Hardware configuration Question #125 also applies to D/T9221. See 'Planning For Your Workstation Subsystem.'

4. 3274 Control Unit, 9370 workstation adaptor

   9370 Work Station Adaptors can have the same problem if all defaults are taken.

   The FILE TRANSFER AID BIT needs to be set in the Work Station Adaptor. See *Planning For Your Workstation Subsystem GA24-4223 p.2-9.* This customer's CE also found that the WORKSTATION CONTROLLER FEATURE 6120 card in the 9370 expansion cage in the 9221 system was not properly configured to operate with 3270 emulation. It must be set to communicate with a 3174 not a 3274.

5. File transfer hang

   Customer was experiencing a file transfer hang at the same exact byte count every time. The problem was intermittent and occurred mostly with larger files. SOLUTION: Hardware Configuration Question #164 PS TERMINAL BIT must be turned off to work correctly.

6. HUNG condition on PC. HANG WAIT or HUNG condition on PC using 4300 ICA and the 3270 Emulation Version 3 Control Program caused by not having EXTDS, Extended Data Streaming in the TCT and VTAM was not configured for DFT Mode.

7. FTTERM configuration problems

   MSGTRANS10 TIMEOUT on PC running FTTERM with a D/T3174 AEA. If the 3174 AEA ports are configured as either FC (FTTERM COLOR) or FM (FTTERM MONO) then answer '1' to 'USE SPECIFIC KEYBOARD MAP (1=YES 0=NO) prompt on the FTTERM configuration. Problems uploading (SEND) files with PC3270 V3.0 attached through a 3708 which uses FTTERM are solved specifying IWS=Y in the setup file xxxxxx.ws under .AEA. See the 3174 Terminal User's Reference. There was no IND$FILE coming in from the emulator on the PC.

8. When RECEIVE is used from PC TRANS10 TIMEOUT 0 BYTES RECEIVED and when you Hotkey back to the host and hit Enter Key, MSGINW0002I INWPRCVE RC1 is issued.

   The TRANS10 was due to not having EXTDS coded. The INW0002I INWPRCVE RC1 was due to Hotkeying back to the and hitting the Enter Key. The problem is resolved by coding EXTDS in the TCT.

9. HANG OR MSGINW0002I INWPGET1 RC=0002 when using PERCOM Personal Communications 3270 V2 under Windows.

   The customer signed on to the IUI and used PF6 to get into native CICS. He opened a window and sent a file to the host. He received MSGINW0001I File Transfer Complete at the PC but it did not show up on the host screen as it does in DOS Mode. He hit enter or PF3 on the host session and INW0002I INWPGET1 RC=0002 was issued. The RC2 means that no acknowledgement was received from the PC. The trace showed normal DFT protocol flow up until the host sent out the Insert Request + Data (msg) and then the PC never responded with an Insert Data acknowledgement positive. The user found that by increasing the TIMEOUT option from 30 to 60 on the windows panel the INW0001I would appear at the host.

10. Timeout on a SNA connected PC. Sending a file to HTF ended in message PCSxfer041 HOST HAS NOT RESPONDED WITHIN TIME-OUT PERIOD. The problem occurred mainly with large files. Increasing the second field of IOAREALENGTH from 4096 to 32K solved the problem. The size specified in the IOAREALENGTH is related to the transfer buffer size in the emulator.

11. Transferring to or from a VSE sublibrary, job PWSLTS is released executing program LIBRLTS to process librarian requests. This job will stay active in case of a connection error covered by the node error program. The job has to be cancelled in such an error situation, otherwise further file transfers to VSE sublibraries will not work.

12. Transferring to or from a VSE sublibrary, job PWSLTS is released executing program LIBRLTS to process librarian requests. This job will stay active in case of a connection error covered by the node error program. The job has to be cancelled in such an error situation, otherwise further file transfers to VSE sublibraries will not work.

13. File transfer using a Telnet attached terminal may terminate with large files. Not message is issued. This is most likely caused by a large packet size specified in the emulator. The specified value for Telnet attached terminals should be smaller than 8000, preferably are 4000.

Return Codes of VSE FILE TRANSFER IWS Messages

Message	Module	RC	Message Explanation
INW0001I	INWPROOT	----	FILE TRANSFER COMPLETE (NORMAL)
INW0002I	INWPCLS0	0001	INVALID REPLY (NO AID ENTER)
	INWPCLS1	0001	NO REPLY RECEIVED
	""	DDM	ERROR REPLY
	INWPGET	0002	LINK TO INWFMGR (RDFILE) FAILED
	INWPGET0	0001	INVALID REPLY (NO AID ENTER)
	INWPGET1	DDM	ERROR REPLY
	""	0001	NO POSITIVE REPLY X'63'
	""	0200	ARRIVAL SEQUENCE INVALID
	""	0001	ISSID NOT EQUAL TO X'63'
	""	0002	NO ACKNOWLEDGEMENT FROM PC
	INWPMSG	????	ERROR OCCURRED BUT MESSAGE COULD NOT BE
	""		DISPLAYED BECAUSE:
	""		1) MESSAGE FILE DISABLED OR NOT OPEN
	""		2) THE MESSAGE WAS NOT FOUND IN FILE
	""		INWPMSG. ???? IS THE HEX NUMBER
	""		OF THE MESSAGE.
	INWPOPN0	0001	INVALID REPLY (NO AID ENTER)
		00FF	DATA SET FULL OR PC ERROR
			PC USER RECEIVES MSGTRANS11
	INWPOPN1	FFFF	NO REPLY RECEIVED FROM OPEN
	""	EEEE	INVALID REPLY FROM OPEN
	""	DDM	ERROR REPLY
	INWPPUT	0001	ERROR DURING DELETEQ TS
	""	0002	LINK TO INWFMGR (STATE) FAILED
	""	0003	LINK TO INWFMGR (WRREP) FAILED
	""	0004	LINK TO INWFMGR (WREND) FAILED
	""	0005	LINK TO INWFMGR (GIVEF) FAILED
	INWPPUT0	0001	INVALID REPLY (NO AID ENTER)
	""	0002	INVALID SEQUENCE NUMBER
	""	0003	INVALID DATA LENGTH RECEIVED
	""	0004	CHECK SUMS DO NOT MATCH
	INWPPUT1	DDM	ERROR REPLY
	""	0001	REPLY NOT 'ACKNOWLEDGE DATA'
	""	5D00	UNSUPPORTED TYPE
	""	0200	ARRIVAL SEQUENCE INVALID
	""	4A00	RECORD LENGTH INVALID
			After APAR PL79999 ABENDAEIV
			may occur see II06689.

Figure 114 (Part 1 of 2). File transfer messages

Message	Module	RC	Message Explanation
	INWPQUER	0001	NON ZERO RETURN CODE ON EXEC
	""		CICS ASSIGN TO DETERMINE IF CU
	""		SUPPORTS EXTENDED DATA STREAM
	""	0002	NO REPLY FROM 'QUERY REPLY'
	""	0003	BAD LENGTH FROM 'QUERY REPLY'
	""	0004	BAD RESPONSE FROM 'QUERY REPLY'
	""	0005	EXEC CICS SEND FAILED
	INWPRCVE	0001	RECEIVE FROM PC FAILED
	INWPREAD	0001	ERROR DURING READQ TS
	""	0002	LINK TO INWFMGR (RDDAT) FAILED
	INWPROOT	0001	CICS HANDLE ABEND ROUTINE CALLED
	""	0002	ERROR RECEIVING INPUT COMMAND
	""	0003	ERROR FROM INWTUID CHECKING UID
	INWPSEND	0001	SEND TO PC FAILED
	INWPWTRE	0001	ERROR DURING READQ TS
	""	0002	ERROR DURING DELETEQ TS
	""	0003	LINK TO INWFMGR (WRDAT) FAILED
	""	0005	ERROR DURING WRITEQ TS
INW0003I	INWPGET	----	GETMAIN REQUEST FAILED
	INWPNIB0		""
	INWPPUT		""
	INWPROOT		""
INW0004I	INWPROOT	----	NO GET OR PUT REQUEST FROM PC
INW0005I	INWPROOT	----	INVALID FILENAME
INW0006I	INWPROOT	----	INVALID OR CONFLICTING OPTION
INW0007I	INWPPUT	----	CANNOT ACCESS HOST TRANSFER FILE
	INWPGET		""
INW0008I	INWPMSG	----	RECORDS SEGMENTED
INW0009I	INWPROOT	----	INVALID USER ID
INW0010I	INWPPUT	----	FILE EXIST, REPLACE NOT ENTERED
INW0011I	INWPPUT	----	NO MORE VSAM SPACE IN XFER
	INWPWRTE		""
INW0012I	INWPGET	----	FILE NOT FOUND
INW0013I	INWPGET	----	FILE NOT SHARED OR PUBLIC
INW0014I	INWPGET	????	UNEXPECTED RETURN CODE ????
	INWPPUT		FROM INWFMGR
	INWPREAD		""
	INWPWRTE		""
INW0015I	INWPPUT	----	DEDICATE NOT SUCCESSFUL
INW0016I	INWPGET	----	ERASE NOT SUCCESSFUL

Figure 114 (Part 2 of 2). File transfer messages

DDM in the return code column indicates a DDM error code listed below:

```
0100 OPEN FAILED                  5800 UNABLE TO FIND FILE
0200 ARRIVAL SEQUENCE INVALID     5D00 UNSUPPORTED TYPE
0300 CLOSE OF UNOPENED FILE       6000 COMMAND SYNTAX ERROR
1A00 FILE NAME INVALID            6200 PARAMETER MISSING
1B00 FILE NOT FOUND               6300 PARAMETER NOT SUPPORTED
1C00 FILE SIZE INVALID            6500 PARAMETER VALUE NOT SUPPORTED
2000 FUNCTION OPEN ERROR          6E00 DATA ELEMENT MISSING
3E00 OPERATION NOT AUTHORIZED     7100 RECORD LENGTH EQUALS ZERO
4700 RECORD NOT ADDED TO FILE     7100 INVALID FLAG VALUE
```

Figure 115. DDM Error Codes

Chapter 18. Interactive Interface, System Files and Configuration

Display VSE Level

The procedure SPLEVEL displays the level of z/VSE, the installation date and the Copyright statement.

```
// EXEC PROC=SPLEVEL
```

Figure 116. Display VSE Level

The SIR command also provides general system information.

Problems Displaying Messages

If a user of the Interactive Interface gets any indication that messages can not be displayed, that file IESTRFL is destroyed or that in error cases there is no message at all, this might be due to the fact that the IESTRFL file got damaged or that it was deleted or that it is not accessible. The text repository file IESTRFL holds dialog messages, explanatory text provided through the Help function and also the text shown in selection panels.

In order to restore and repair the file, skeleton SKRSTRFL in ICCF library 59 can be used.

On older systems, an easy way to repair the file would be to install a PTF which services member IESTRFL.V, the IBM support group can certainly help you identifying such a PTF fitting to your system. A second possibility is to restore member IESTRFL.V from the installation tape and run following job:

```
* $$ JOB JNM=TRFL,CLASS=0,DISP=D
// JOB TRFL CATALOG  TO TEXT REPOSITORY FILE
* CEMT SET DA(IESTRFL) CLO
// PAUSE CLOSE IESTRFL (CEMT SET DA(IESTRFL) CLO
// UPSI 0
// DLBL IESTRFL,'VSE.TEXT.REPSTORY.FILE',,VSAM,CAT=VSESPUC
// EXEC IESTRFUT
* $$ SLI MEM=IESTRFL.V
/*
// PAUSE OPEN  IESTRFL (CEMT SET DA(IESTRFL) OPE
/&
* $$ EOJ
```

Figure 117. How to Get IESTRFL on a VSE System

The IESTRFL.V member is on the NLS library on the tape or cartridge.

History File Damaged

It is not quite easy to find out if the history file is damaged. An indication would be, for example, if service is shown as not installed but definitely was installed. There are also other possible hints for a defective history file like message M432I, in any case IBM should be contacted for a detailed analysis. Some OEM products establishing workfiles do generate message M432I. It is recommended to run MSHP, especially the repair job below, with this products disabled for that partition.

The following sample job is for previous z/VSE releases.

From z/VSE 4.1 on you can use the dialog *Defragmentation of History File*.

Following is a sample job which repairs the internal structure of the history file by means of the merge function. This job may be used with a corrupted history file as well as a good one, it will be reorganized and compressed, so that it might also improve performance.

The job does first copy the history file to an auxiliary history file and afterwards merges the auxiliary back to the system history file. It is recommended to backup the history file before you run this job using the MSHP BACKUP HISTORY function.

```
// JOB RESTORE
// DLBL IJSYS02,'WORK.HIST.FILE'
// EXTENT SYS018,SYSWK1,1,0,960,75
// ASSGN SYS018,DISK,VOL=SYSWK1,SHR
* CHECK THE ABOVE DLBL IF IT REFLECTS THE WORK HISTORY FILE ON
* YOUR SYSTEM, IT IS SET UP FOR 3380'S.
* WE SUGGEST TO BACKUP THE HISTORY FILE BEFORE YOU RUN THIS JOB.
// PAUSE
// EXEC MSHP
CREATE HIST AUX
COPY   HIST SYS AUX
CREATE HIST SYS
MERGE  HIST AUX SYS
/*
/&
```

Figure 118. Restore History file

Mismatch History File - Dialogs

The System History File holds information about the parts contained in your system and the service applied to them.

Some dialogs, especially the IBM Service Dialogs, do not work on the system history file directly, but on the table DTRIHIST.Z in IJSYSRS.SYSLIB. Each time a job created by a dialog changes the history file, the table DTRIHIST.Z is updated automatically by program DTRIPST.

In case, the history file was changed MSHP commands directly, the updates to the table DTRIHIST.Z are missing. Then the two sources of information reflect different states.

In that case, run DTRIPST to update DTRIHIST.Z out of the system history file.

```
// JOB UPDATE
* THIS JOB REFRESHES THE CONTENTS OF DTRIHIST.Z
* USING DATA COMING FROM THE SYSTEM HISTORY FILE
// EXEC DTRIPST,SIZE=250K
/*
/&
```

Figure 119. Update DTRIHIST.Z

Missing History File Contents

When you use the service dialog and get the message

```
BASUC missing
```

this can have the following reason.

Some customers use the system history file for IBM products and a second history file for vendor products.
The service dialogs access the history file over IJSYSHF and use DTRIPST to update DTRIHIST.Z.
If service to a product in the second history file is done using the dialogs, the work history file is mixed up and products entries in the system history file may be deleted.
This situation can be clearified by a retrace of both history files; the system history file must then be repaired.

Restore of the Online Messages Explanation (OME) File

In case the online messages explanation file IESMSGS is damaged, it can be restored from the installation tape using skeleton SKOMERST in ICCF library 59.

Activate TCP/IP messages in the Online Message File

The Online Messages Explanation file does not contain TCP/IP messages. If you want to use the online message explanation you can update the OME with these messages. TCP/IP provides a job named IPNOME in library PRD2.TCPIPC. Simply run this job, for example as listed in the following job.

```
* $$ JOB JNM=IPNOME,DISP=D,PRI=9,NTFY=YES,CLASS=0
* $$ LST DISP=H
// JOB IPNOME    UPDATE THE OME WITH TCP/IP MESSAGES
* $$ SLI MEM=IPNOME.Z,S=PRD2.TCPIPC
/*
/&
* $$ EOJ
```

Message: User ID Already in Use

If this message occurs, in many cases it is caused by the fact that the related terminal used by this user got a connection error or was switched off without performing sign-off. It is also issued in case the user is still logged on at another terminal. In order to sign-off automatically from VSE/ICCF and also from the z/VSE Interactive Interface in case of an error, the system supplied node error program IESZNEP should be activated. IESZNEP is activated by default for VSE/ESA 2.4.0 or later. There is more information provided about this program in the z/VSE Administration manual.

With VSE/ESA 2.4.0 and later, you may use PF12 after you get message "User ID Already in Use" for the LOGON HERE function. You will not get PF12 in case you have set security off in the SIT.

How to Syncronize System Data Which are Based on User-IDs

The dialog "Maintain User Profiles" (FP 211) provides adding, changing or deleting user-IDs in z/VSE. At the same time these user-IDs are used to manage user connects to groups via the dialog "Maintain Security Profiles" (FP 282). Additionally the same user-IDs are used in the dialog "Maintain LDAP User Prifiles" (FP 217) to control the LDAP profiles if LDAP is enabled in the system.

Before the customer was suggested to synchronize the data across all three dialogs via the manual calls of them. Starting with z/VSE 4.3 each time when a user-ID is added, changed, or deleted in the dialog "Maintain User Profiles" (FP 211), then the customer is passed through the dialogs "Maintain Security Profiles" (FP 282) and "Maintain LDAP User Profiles" (FP 217) automatically to synchronize the system data which are based on user-IDs if required. The corresponded system data are NOT changed automatically, only one of the possible synchronization scenarious is supported by IUI.

For each of three well known types of user-IDs (SYSA, PROG, and OPER) and for each of three UID options (1=ADD, 2=CHANGE, 5=DELETE) IUI provides a special synchronization scenario:

Options in Panel 211 (UIDs)	Option 1=ADD		Option 2=CHANGE		Option 5=DELETE
Types of User-ID	SYSA	PROG OPER	SYSA	PROG OPER	Any User-ID
Call Panel 217 (LDAP)	YES	YES	YES	YES	YES
Call Panel 282 (BSM)	NO	YES	NO	YES	YES

In case of problems with the dialog dialog "Maintain User Profiles" (FP 211) you should check if the dialogs "Maintain LDAP User Profiles" (FP 217) and "BSM Group Maintenance" (FP 282) work without errormessage. Problems could be missing CICS or security definitions.

Deactivate the z/VSE Interactive Interface

The interactive interface is started by program IESCICIN which is activated at PLT startup time through the DFHPLTPI phase. In order to deactivate the interactive interface, the related entry for IESCICIN should be deleted in the IESZPLTI copy book which is used in DFHPLTI. If VSE/ICCF should also be deactivated, entry DTSPOSTI should also be deleted. During shutdown, the interactive interface is terminated through entry IESCICSD in copy book IESZPLTS. This copy book is used in the DFHPLTSD shutdown program. Again, if VSE/ICCF is also not used, entries DTSICCF and DTSSHUT should also be eliminated. The default DCT needs also to be changed - the indirect destinations IESN and IESM should not be used; that is, do not use the copy book IESZDCT, and also do not trigger transaction IEWR for destination IEP1.

Note: Deactivating the interactive interface implies that not only the dialogs will not work but also sign on routine IESxLOGO and functions like the file transfer to and from workstations or the online problem determination program. How to use the file transfer without the interactive interface is described in the *Programming and Workstation Guide*.

Problems Using the Dialog "Storage Dump Management"

With the storage dump management dialog, Info/Analysis is run in the VSE/ICCF interactive partition to retrieve dump information from the dump management file. Dumps can be stored in the system dump library or, starting with VSE/ESA 3.1, in the dump archive. The dump archive is a sublibrary named PRD2.DUMP located in VSAM space, having the advantage that it is enlarged automatically whenever more space is needed. To improve performance, it is recommended to delete obsolete dumps from the library.

The dialog offers the key PF9=DEL_ALL to allow deleting groups of dumps using the program DMPMGR (see “REXX program DMPMGR to Manage the Dump Library” on page 302)

If the dialog does not work, following steps may help finding the reason for the problem. If messages indicate that there is not enough GETVIS available, or in any other problem case, check the following:

1. Does following job run in batch:

```
* $$ JOB JNM=DUMPLIST,DISP=D,CLASS=A,PRI=3
// JOB DUMPLIST  PRINT COMPLETE FORMATTED DUMP
// EXEC PROC=DTRINFOA
// EXEC INFOANA,SIZE=300K
   SELECT DUMP MANAGEMENT
     PRINT DATA
          RETURN
   SELECT END
/*
/&
* $$ EOJ
```

Figure 120. Info/Analysis Dump Sample Job

2. If the job runs in batch, it will show you the dumps stored in the dump library, you could also use the batch interface as a circumvention to print off a certain dump or to delete dumps.
3. If the job does not run in batch, check the messages you get from Info/Analysis in the *Messages and Codes* manual. The problem might be due to Info/Analysis dump file being corrupted. In order to repair the dump management file, you should reformat it and initialize it. Before formatting, it can be scratched using DITTO - this will ensure the file is in fact cleared. To redefine the dump files, use skeleton SKDMPINI in library 59. For releases prior to z/VSE 3.1.0, scratching the dump management file is recommended before running the job. Scratching is done using DITTO:

```
* $$ JOB JNM=SCRATCH,DISP=D,CLASS=A,PRI=3
// JOB SCRATCH DUMP MANAGENET FILE
// UPSI 1
// EXEC DITTO
$$DITTO PVT INPUT=VSYSWK1,SCRATCH=INFO.ANALYSIS.DUMP.MGNT.FILE
/*
/&
* $$ EOJ
```

Figure 121. DITTO Scratching the Dump Management File

Note that problems might occur if dumps of former releases are still in the dump library.

If problems persist: contact Info/Ana-Level2.

If the batch job worked but the dialog did not:

4. Did the dialog abend ? Which messages were issued ?
5. Check library member PRB$TEM2 and/or PRB$TEM3 in your primary ICCF library. If members PRB$TEM2 and/or PRB$TEM3 are present,
6. Check size and GETVIS of interactive partition (/MAP on console); It should be at least 768K (or 1024K).
7. Check size and GETVIS of CICSICCF partition (default is F2; MAP F2 or GETVIS F2). Free GETVIS should be at least 150K. Free GETVIS must be below 16 MB.
8. Was the size of the Info/Analysis file increased ? E.g. BLN2013I Dump management file error. Reason=2008. The file is shipped on 3380 disks with 3 tracks on 2.3.0 or higher.

```
// DLBL BLNDMF,'INFO.ANALYSIS.DUMP.MGNT.FILE',0
// EXTENT SYS016,SYSWK1,1,0,9030,3
// DLBL BLNXTRN,'INFO.ANALYSIS.EXT.RTNS.FILE',1999/365,SD
// EXTENT SYS017,SYSWK1,1,0,9045,1
```

Figure 122. Dump Management File

9. If message PRB$TEM1 not found occurs, you should update the system definitions for Selection Panels and Application Profiles. You can do this by logging on with users SYSA and pressing PF6 on fastpath 212 and fastpath 213.

Problems Using the Dialog "Maintain LDAP User Profiles"

To be able to use the dialog Maintain LDAP User Profiles the following preparation must be done.

- The LDAP service must be configured, this means the LDAP configuration phase IESLDCFG must exist in PRD2.CONFIG.

If this is not the case, the following message is shown and the dialog is ended.

```
LDAP SERVICE IS NOT CONFIGURED. USE SKELETON SKLDCFG IN ICCF 59 TO DO SO
```

If you run the system in basic mode you get the same message, although the LDAP service has already been configured. The reason is, that the library PRD2.CONFIG is not in the search chain for the basic CICS partition.

Abend DM02

The dialog manager frequently abends with an abend DM02. In most cases, the reason for DM02 is that the primary library has reached the maximum number of members allowed. If the abend happens, a panel as following is displayed:

```
 ...+....1....+....2....+....3....+....4....+....5....+....6. <==MORE==> .+..RD
INTERACTIVE INTERFACE ABEND CODE: DM02
0S19I OPERATOR/ICCF SYSTEM REQUEST
0S00I SUB DTRDDMGR CANCELED
0S07I PROBLEM PROGRAM  PSW = 07DD2000 00754022
*VSE ICCF PARTITION DUMP PROGRAM
K402I  CSECT ENTERED = DTRDDMGR
K403I  ICCF RETURN CODE = 07
K401I     LOAD HI-PTN HI-PHS ORIGIN  SCAN*   ***STATUS***
ACTUAL: 7523C0 851FFF 78032F 7523C0 7523C0
RLATIV: 000000 0FFC3F 02DF6F 000000 000000
K442I  PSW=07DD200000754022
K441I  ACTUAL=754020  REL=001C60
K443I  INSTR=0A0441E10008
    .
GP 0-F 00000000 00754028 0079D0D8 00000000 0079D0A6 0079B09A 0079B09A 80753EC8
       00000000 007A0A64 0079D000 007A0B0C 60753BC2 0079D058 B0754014 BF0003E7
K404D  ENTER DUMP COMMAND
*ENTER DATA?
```

The Interactive Interface abend codes are described in the message manual section IICO.

As a quick solution for the problem you may delete some members in your primary library, or you may extend the maximum number of entries in your primary library. To do so, run $DTSUTIL in ICCF command mode (fastpath 6) and perform command ALTER LIB(nn) MAXDIR(300) this would extend to a maximum of 300 entries.

Abend ST01

The dialog manager may abend with an abend ST01 when using specific dialogs. The reason for ST01 is that internal tables are too huge for processing.

This may be the case in a system with many hardware devices resulting in huge hardware tables.

In this case there is a need to change the size of your interactive partitions, you must tailor VSE/ICCF with new sizes. To perform this task you can either use

- Skeleton SKICFGEN in ICCF library 59
- Activate the optional VSE/ICCF generation phase DTSIGENM, which is provided in IJSYSRS.SYSLIB. To do so, end VSE/ICCF by entering /ICCFEND then start ICCF again with the I$ST DTSGENM command.

The Interactive Interface abend codes are described in the message manual section IICO.

Old CSD File

Some users restored the CICS CSD file from an old system after installing a new release. This leads to messages PROGRAM NOT DEFINED TO CICS. For VSE/ESA 2.3.0 and higher skeleton SKCSDFIL in ICCF library 59 can be used to upgrade the IBM provided CSD settings like for CICS, Interactive Interface and ICCF.

In case of VSE/ESA 2.4.0 and later, it is not possible to copy an old level CSD file, the CSD file needs to be migrated from CICS/VSE to CICS TS. It is recommended to use the newly created CSD file and add definitions to it with the help of DFHCSDUP or the migration of PPTs and PCTs. In case a secondary CICS TS is defined, related definitions must also be established. Refer to SKPREPC2 for more information.

If a Fast Service Upgrade (FSU) is performed (including a version upgrade), the CSD file for parts related to DBDCCICS is upgraded through the FSU process in FSU step 4C.

For the production CICS PRODCICS, the VSELST2 needs also to be updated in case of an FSU from 4.3.x or 5.1.x to z/VSE 5.2. Skeleton SKCSDFC2 in ICCF library 59 can be used.

CICS TS CSD Migration

Summary

Best practice to define user supplied resources to CICS TS CSD file is to have a DFHCSDUP job to define your CSD resources. Having existing CSD definitions (and lacking a definition job) a possible way is to extract the existing profiles from the CSD and create a defintion job out of it. This doc gives a step-by-step description how to do this.
Use sample JCL SKCSDFC2 to integrate the created .Z book with the definition statements of your resources. Run this JCL after re-allocating the CICS CSD file (e.g after initial install or FSU).

For more information migrating CICS CSD file in general see *CICS TS Enhancements Guide*.

Extract profiles from CICS TS CSD file

The DFHCSDUP commands to extract profiles are:

```
// EXEC DFHCSDUP,SIZE=800K,OS390       USE CSD BATCH UTIL PROGRAM
   EXTRACT  GROUP(VSESPG)  OBJECTS USERPROGRAM(DFH0CBDC)
/*
```

This requires a USERPROGRAM/EXIT to store the data in a file.

Establish USERPROGRAM DFH0CBDC

1. Pre-req: COBOL compiler
2. Use source/sample program DFH0CBDC.C from PRD1.BASE
3. For using IUI compile function, copy into ICCF library using

   ```
   LIBRP PRD1.BASE DFH0CBDC.C DFH0CBDC
   ```

   Drop catalog and end of catalog statements from ICCF member
4. Use IUI/ICCF compile function with

   ```
   Source Type=2 (Batch Program)
   Language=3 (COBOL VSE)
   Catalog=1 (Yes)
   Output Member=COMCBDC
   ```
5. Update compile/catalog JCL
   - specify sublib for phase catalog
   - INCLUDE DFHEXCI
   - ENTRY DFHEXTRA

For more information on using DFHCSDUP with a USERPROGRAM exit see *CICS Customization Guide* section "DFHCSDUP as a batch program".

The complete compile/catalog JCL:

```
* $$ JOB JNM=COMWACK,DISP=D,CLASS=A,NTFY=YES
* $$ LST DISP=D,CLASS=Q,PRI=3
// JOB COMWACK COMPILE PROGRAM DFH0CBDC
// LIBDEF *,SEARCH=(PRD2.SCEEBASE,PRD2.PROD)
// SETPARM CATALOG=1
// IF CATALOG = 1 THEN
// GOTO CAT
// OPTION ERRS,SXREF,SYM,LIST,NODECK
// GOTO ENDCAT
/. CAT
// LIBDEF PHASE,CATALOG=PRD2.CONFIG
// OPTION ERRS,SXREF,SYM,NODECK,CATAL
   PHASE DFH0CBDC,*
   INCLUDE DFHEXCI
/. ENDCAT
// EXEC IGYCRCTL,SIZE=IGYCRCTL
 CBL LIB,APOST,NOADV,RENT,BUF(4096)
* $$ SLI ICCF=(DFH0CBDC),LIB=(0010)
/*
  ENTRY DFHEXTRA
// IF CATALOG NE 1 OR $MRC GT 4 THEN
// GOTO NOLNK
// EXEC LNKEDT,SIZE=256K
/. NOLNK
/&
* $$ EOJ
```

Figure 123. DFHCSDUP compile job sample

Allocate output file

An output file is required to store the extracted profile data. Use CYL=20 (3390).

Run the DFHCSDUP EXTRACT JCL

```
* $$ JOB JNM=CSDEXTRA,CLASS=0,DISP=D,PRI=3
// JOB CSDEXTRA  EXTRACT FROM CSDFILE
* ********************************************************************
*
*  ---------------- C I C S   CSD EXTRACT      ----------------------
*
*  THIS SKELETON MAY BE USED TO PERFORM AN EXTRACT FROM
*  CICS CSD FILE FOR YOUR SECOND CICS
*
* ********************************************************************
*
*   CLOSE DFHCSD FILE FIRST
*
*   ISSUE FOLLOWING COMMANDS SEQUENCE ON THE CONSOLE AFTER  // PAUSE
*     MSG F2
*     XX CEMT SET FI(DFHCSD) CLOSE    (XX = REPLY-ID)
*     REPLY  "(END/ENTER)"  TO CONTINUE
* --------------------------------------------------------------------
// PAUSE
// DLBL CBDOUT,'CICS.DFHCSD.OUTPUT',0,SD
* ADJUST EXTENT INFO TO YOUR ENVIRONEMNT
// EXTENT SYS002,SYSWK1,1,0,111510,300
// ASSGN SYS002,DISK,VOL=SYSWK1,SHR
// LIBDEF *,SEARCH=(PRD2.CONFIG,PRD1.BASE,PRD2.SCEEBASE)
// EXEC DFHCSDUP,SIZE=800K,OS390       INIT AND LOAD CICS
  EXTRACT  GROUP(VSESPG)  OBJECTS USERPROGRAM(DFH0CBDC)
/*
/&
* $$ EOJ
```

Figure 124. DFHCSDUP Extract from CSD file

Get statements from output file

To get DFHCSDUP output into user accessible format, the extract output can be copied into a LIBR member using DITTO:

```
* $$ JOB JNM=DITEXTPR,DISP=D,CLASS=0
// JOB DFHDITPR         PRINT EXTRACTED DATA
// UPSI 1
// DLBL CBDOUT,'CICS.DFHCSD.OUTPUT',0,SD
// EXTENT SYS019,SYSWK1,1,0,111510,300
// ASSGN SYS019,DISK,VOL=SYSWK1,SHR
// EXEC DITTO
* DITTO SP FILEIN=CBDOUT
$$DITTO SL FILEIN=CBDOUT,LIBOUT=PRD2.CONFIG,MEMBEROUT=CSD0EXT.Z
$$DITTO EOJ
/*
/&
* $$ EOJ
```

Figure 125. Print DFHCSDUP extracted data

IPF Error

When the II dialog manager returns to the Service Manager, the Service Manager checks for an error return code. If there is an error, you may receive the following message on your terminal:

```
THE PROGRAM YOU WERE USING ENDED ABNORMALLY.
PRESS PF1 FOR MORE INFORMATION
```

The corresponding help gives the following information:

```
         Up to 20 lines of information about the error were placed
         in the online system's error log.

         There may be more than 20 lines of information available.
         If you want more information about the error, library member
         DTRLxxxx (xxxx = your User ID) should have been created in
         the ICCF library which was the primary library at the time
         the program was called.  This may be the library assigned as
         your primary library or one you had previously switched to.

         Edit (or list) member DTRLxxxx to get more information.

                          PF3=END
```

Please check the member DTRLxxx in your primary ICCF library. This member looks like the following example.

```
--------------------------------------------------------------------------
DISPLAY  COM$ICL 0146 DTRSDSP  DP02                        COM$ICL1
--------------------------------------------------------------------------
  DP02 - PANEL FILE NOT FOUND OR A VALID HEADER RECORD COULD NOT
         BE FOUND. THE NAME OF THE PANEL FILE IN ERROR IS IN
         UCAERFN.

MEMBER DTRMSG OF LIBRARY 2 SHOWS THE FORMAT OF THE ABOVE RECORD.
```

The format of the error record is as follows:

```
*********************************************************************
*                 FORMAT OF ERROR LOG ENTRY                         *
*********************************************************************
    FIELD DESCRIPTION                      OFFSET         FIELD NAME
NAME OF CURRENT SERVICE                     1-8            UCASERV
NAME OF INVOKING FUNCTION                  10-17           UCAFUNCT
LINE NUMBER OF INVOKING FUNCTION           18-21           UCAERLNO
NAME OF ROUTINE DETECTING ERROR            23-30           UCAERRTN
ERROR CODE UNIQUE TO DIALOGUE MANAGER      32-35           UCAERCD
CURRENT LINE NUMBER OF DISPLAY PANEL       37-38           UCAERCL
OR HEXADECIMAL OFFSET WITHIN MODULE        37-39
CURRENT COLUMN NUMBER OF DISPLAY PANEL     40-41           UCAERCC
INFORMATION ASSOCIATED WITH A MESSAGE:     43-50           UCAERVV
VARIABLE VALUE, VARIABLE NAME, MESSAGE
IDENTIFICATION ETC.
CURRENT RECORD NUMBER OF FILE IN ERROR     51-54           UCAERCR
ERROR CODE OF INTERACTIVE SUBSYSTEM        56-59           UCAERSYS
FILE NAME OF FILE IN ERROR                 61-68           UCAERFN
FILE TYPE OF FILE IN ERROR                 70-77           UCAERFT
FILE MODE IF CMS IMPLEMENTATION            79-80           UCAERFM
*********************************************************************
```

How to Find System Member DTR$DTBL in ICCF Library

Member DTR$DTBL in ICCF library 58 is a central IPF control member. Here you can find synonym tables and PF-key tables.

The member tells in which ICCF library panels or tables with a specific prefix can be found. Also the ICCF library number for a specific system member can be found here.

Hints and Tips for Fast Service Upgrade (FSU)

FSU to z/VSE 6.2 is possible from z/VSE 6.1 only.

FSU to z/VSE 6.1 is not possible.

General Information for a Version Upgrade

The following version upgrades are possible:

```
Older Versions:
     From VSE/ESA 1.3.x --> VSE/ESA 2.1.x
                        --> VSE/ESA 2.2.x
                        --> VSE/ESA 2.3.x
     From VSE/ESA 1.4.x --> VSE/ESA 2.1.x
                        --> VSE/ESA 2.2.x
                        --> VSE/ESA 2.3.x
     From VSE/ESA 2.1.x --> VSE/ESA 2.3.x
     From VSE/ESA 2.2.x --> VSE/ESA 2.3.x

     From VSE/ESA 2.4.x --> VSE/ESA 2.5.x
     From VSE/ESA 2.4.x --> VSE/ESA 2.6.x
     From VSE/ESA 2.5.x --> VSE/ESA 2.6.x

     From VSE/ESA 2.5.x --> VSE/ESA 2.7.x
     From VSE/ESA 2.6.x --> VSE/ESA 2.7.x

     From VSE/ESA 2.6.x --> z/VSE 3.1.x
     From VSE/ESA 2.7.x --> z/VSE 3.1.x

     From VSE/ESA 2.7.x --> z/VSE 4.1.x
     From VSE/ESA 3.1.x --> z/VSE 4.1.x

     From VSE/ESA 3.1.x --> z/VSE 4.2.x
     From VSE/ESA 4.1.x --> z/VSE 4.2.x

     From VSE/ESA 4.1.x --> z/VSE 4.3.x
     From VSE/ESA 4.2.x --> z/VSE 4.3.x
     From VSE/ESA 4.2.x --> z/VSE 5.1.x
     From VSE/ESA 4.3.x --> z/VSE 5.1.x
     From VSE/ESA 4.3.x --> z/VSE 5.2.x
     From VSE/ESA 5.1.x --> z/VSE 5.2.x
Current Versions:
     From z/VSE 6.1.x --> z/VSE 6.1.y
     No FSU from older z/VSE versions possible.
     From z/VSE 6.1.x --> z/VSE 6.2.x
     From z/VSE 6.2.x --> z/VSE 6.2.y
```

Note: VSE/ESA 2.6, VSE/ESA 2.7 and z/VSE 3.1 cannot be installed via FSU (version upgrade) from a system prior to 2.4.0; an initial installation is required to upgrade from such a system.

```
* ------------------------------------------------------------------
* STEP  2: RESTORE FSU PROGRAM SRV$FSU INTO IJSYSRS.SYSLIB
* ------------------------------------------------------------------
* 1) MOUNT z/VSE5.X.X-YY ON TAPE-DRIVE 182
* AND REPLY "(END/ENTER)" TO CONTINUE
* ------------------------------------------------------------------
// PAUSE
// EXEC DTRSETP,SIZE=AUTO,PARM='SRV$SYS;PRD2.CONFIG;SET XLAST=P02'
/*
// OPTION IGNLOCK
// ASSGN SYS005,182
// MTC REW,SYS005
// EXEC LIBR,PARM='MSHP'
RESTORE IJSYSR1.SYSLIB.SRV$FSU.PHASE : IJSYSRS.SYSLIB -
               LIST = NO -
               REPLACE = YES -
               TAPE = SYS005
/*
```

Figure 126. FSU Preparation Job

For a release upgrade in general, down level check should not be used.

The FSU dialog supports

- a real tape drive
- a virtual tape on VSAM

Not supported by the dialog are

- FSU from installation disk (see Chapter 25, “z/VSE Initial Installation using an Installation Disk” on page 363). This is a permanent restriction.
- tapes managed by a tape server

To use tapes managed by a tape server you can issue a LIBSERV MOUNT AR-type cmd (which includes the partition accessing the tape) from the console prior to executing the FSU jobs and issue a LIBSERV EJECT after the FSU job has finished. Of course you have to repeat the LIBSERV MOUNT after IPL for stage 2.

FSU Requirements

Performing an FSU is an easy way to upgrade the system. Basically there are two types of FSU, a normal refresh installation for example like an FSU from 5.2.0 to 5.2.1 and a so called version or release upgrade, for example from 5.1.0 to z/VSE 5.2.0. A release upgrade needs detailed planning for several reasons like system layout changes and similar topics. A service refresh normally does not change any storage sizes or layout specific things. For more details refer to the Planning Guide and also to the System Upgrade and Service manual. Here just some remarks regarding version and release upgrades:

- Upgrading from 2.6.x to 3.1.x needs additional space for PRD2 library (about 10000 library blocks). This space has to be available for the master catalog. For file BSTCNTL, additionally about 2 cylinders (3390) are requested in the user catalog.
- Upgrading from 2.7.x to 3.1.x needs additional space for PRD2 library (about 4000 library blocks). This space has to be available for the master catalog. For file BSTCNTL, additionally about 2 cylinders (3390) are requested in the user catalog.

- FSU from releases prior to z/VSE 3.1.0 is not possible if the system packs DOSRES and SYSWK1 are FBA devices or 9345 DASDs.
- An upgrade from VSE/ESA 2.7 will not require a VSE/POWER cold start, however, the POWER files will be migrated to the new format. After migrating, it is not possible to start the old system from DOSRES. It is recommended to save the POWER files at the end of Stage 1 of the FSU. The following message is issued during Stage 2 of FSU:

  ```
  1Q0HD IF SPOOL FILE MIGRATION TO V9R2 IS INTENDED REPLY 'YES',
  ELSE 'NO'
  ```

  Please enter 'YES' to have the queue files converted. In case your system is started usually with security set on, the following message is shown:

  ```
  1QFFD VSE/POWER WARMSTART AND VSE ACCESS CONTROL NOT ACTIVATED
  (SEC=NO). DO YOU WISH TO CONTINUE? (YES/NO)
  ```

  Please enter 'YES' to continue. If you enter 'NO', the system will stop.
- The FSU requires VSESPUC to be defined on DOSRES and SYSWK1. If you moved the VSESPUC to a different volume, or you removed the space on DOSRES or on SYSWK1, you may face problems during the FSU.
  The jobs DTRFSU15 and DTRFSU4C define new system files. These files are located in VSESPUC, the volumes DOSRES and SYSWK1 are to contain the cluster.
  If your VSESPUC does not contain space on DOSRES and SYSWK1, you have to edit the FSU jobstream and update the VOLUMES(DOSRES,SYSWK1) parameter in DTRFSU15 and DTRFSU4C.

The FSU has the following general requirements:

- The system volumes DOSRES and SYSWK1 exist.
- The z/VSE library structure must exist. This means that predefined libraries and sub-libraries like IJSYSRS.SYSLIB, PRD1, PRD2.CONFIG, PRD2.SCEEBASE, PRD2.SAVE and other exist.
- The VSAM master catalog IJSYSCT VSAM.MASTER.CATALOG and the user catalog VSESPUC VSESP.USER.CATALOG exist with these file IDs and file names.
- System files like the VSE Control File or the Text Repository File exist

If your system does not have the provided system layout, if you have moved or removed any of the system files or libraries, you should be aware that the FSU relies on this information.

Upgrade to z/VSE 5.1

- Check the ALLOC procedure before the FSU is done. Make sure that the partition FB has at least 1 MB allocated. You can use SKALLOC* for reference.
- To avoid the message *M063I Insufficient storage in partition* consider to use the dialog *147 Defragmentation of History File* .
- The z/VSE 5.1 needs about 4000 library blocks more (PRD1) than the z/VSE 4.1 system needed. Concerning storage requirements from VSE/ESA 4.1.x see System Upgrade and Service manual, in addition you have to consider above 4000 blocks. The storage needs to be defined in the master catalog.

- Since z/VSE 5.1 is shipped with supervisor $$A$SUPI only, at the end of stage 1, you may generate your own supervisor, however, the supervisor generation will not allow to select the supervisor name anymore. You also just may change the supervisor name during Stage 2 to the default name $$A$SUPI.
- On z/VSE 5.1, label area is only possible on a virtual disk. If you still have a DLA command in your IPL procedure, the DLA command will be ignored, but there is a // EXEC PROC=STDLABEL statement before the VDISK command. This statement has to be removed in $0JCL procedure before Stage 1 of the FSU is started. $0JCL has to be changed in PRD2.SAVE as well. The procedure should look as follows - with only one call of the label procedure:

  ```
   ...
  SYSDEF DSPACE,DSIZE=15M
  // VDISK UNIT=FDF,BLKS=2880,VOLID=VDIDLA,USAGE=DLA
  // EXEC PROC=STDLABEL                    LOAD LABEL AREA VDISK
   ...
  ```

- In case you did not modify the $0JCL procedure, especially if not modified in PRD2.SAVE, You will get following message in Stage 2 of the FSU:

  ```
  1L41D  LABEL AREA NOT DEFINED
  ```

  You may continue entering:

  ```
  0 // VDISK UNIT=FDF,BLKS=2880,VOLID=VDIDLA,USAGE=DLA
  0 // OPTION STDLABEL
  0
  ```

  System continues the startup and will issue message:

  ```
  1UV9D LABEL AREA IS ALREADY ON VIRTUAL DISK FDF
  ```

  on the second VDISK command. This message can be ignored.
- The node error program IESZNEP needs to be re-compiled, use skeleton in ICCF library 59.
- Remove the *// EXEC IESIRCVT* statement from the USERBG procedure, use skeleton SKUSERBG in ICCF library 59.
- An upgrade to z/VSE 5.1 will not require a VSE/POWER cold start, however, the POWER files will be migrated to the new format. After migrating, it is not possible to start the old system from DOSRES. It is recommended to save the POWER files at the end of Stage 1 of the FSU. The following message is issued during Stage 2 of FSU:

  ```
  1Q0HD IF SPOOL FILE MIGRATION TO V9R1 IS INTENDED REPLY 'YES',
  ELSE 'NO'
  ```

  Please enter 'YES' to have the queue files converted. In case your system is started usually with security set on, the following message is shown:

  ```
  1QFFD VSE/POWER WARMSTART AND VSE ACCESS CONTROL NOT ACTIVATED
  (SEC=NO). DO YOU WISH TO CONTINUE? (YES/NO)
  ```

  Please enter 'YES' to continue. If you enter 'NO', the system will stop.

Check the dialog 141 Verify Location of Involved Serviced Files if your change is supported here and change the location prior to the FSU dialog.
Check the created FSU job, if it meets your needs.

Upgrade to z/VSE 5.2

- An upgrade to z/VSE 5.2 will not require a VSE/POWER cold start, however, the POWER files will be migrated to the new format. After migrating, it is not possible to start the old system from DOSRES. It is recommended to save the POWER files at the end of Stage 1 of the FSU. The following message is issued during Stage 2 of FSU:

  ```
  1Q0HD IF SPOOL FILE MIGRATION TO V9.2 IS INTENDED REPLY 'YES',
  ELSE 'NO'
  ```

  Please enter 'YES' to have the queue files converted. In case your system is started usually with security set on, the following message is shown:

  ```
  1QFFD VSE/POWER WARMSTART AND VSE ACCESS CONTROL NOT ACTIVATED
  (SEC=NO). DO YOU WISH TO CONTINUE? (YES/NO)
  ```

  Please enter 'YES' to continue. If you enter 'NO', the system will stop.

- The FSU automatically defragments the system history file in job DTRFSU14.

Upgrade to z/VSE 6.1

No FSU is possible.

Upgrade to z/VSE 6.2

FSU is possible from z/VSE 6.1.

What To Do if Errors Occur

The System Upgrade and Service manual shows a couple of things that can help to solve problems during FSU. Since the FSU process changed with the new tape layout introduced with 2.7.0, not all steps perform the same function as in previous releases, so for example step 23 is now installing the LE code, in former releases it installed PRD1. Here are some other tips or more general nature:

- Job DTRFSU02 fails executing the Preparation job sequence. Message

  ```
  L054I SPECIFIED BACKUP FILE ID NOT FOUND ON INPUT TAPE
  ```

  is issued. This is due to the fact that the Preparation job has to be changed, see section above.

- Invalid allocation message. Probably PASIZE is not correct. This should occur only for release or version upgrades.

- wrong allocation for FB The partition FB for the security manager must have at least 1 MB allocated. You can use SKALLOC* for reference.

- DLA statement in IPL procedure If your IPL procedure contains a DLA statement you get the message:

  ```
  1L41D  LABEL AREA NOT DEFINED
  ```

  You may continue entering:

  ```
  0 // VDISK UNIT=FDF,BLKS=2880,VOLID=VDIDLA,USAGE=DLA
  0 // OPTION STDLABEL
  0
  ```

  System continues the startup and will issue message:

  ```
  1UV9D LABEL AREA IS ALREADY ON VIRTUAL DISK FDF
  ```

- Wrong disk IPLed during Stage 2. In most cases the FSU will fail during step DTRFSU4C with message MEMBER IESTRFL.V not found. It may also fail

earlier with message PROCEDURE $0JCLFSU not found. If it does not fail in DTRFSU4C it will fail in step DTRFSU26, you will get a message that library IJSYSRS is in use. You should IPL from SYSWK1 and rerun the whole stage 2 or reset to the appropriate job.

- POWER does not come up during Stage 2. You should check for old level POWER phases in sublibrary PRD2.SAVE. Such a phase would be copied into IJSYSR1.SYSLIB and get activated.
- IPL does not work during Stage 2. This may be due to similar reason as last point, old level system phases should not be stored in PRD2.SAVE.
- POWER files are migrated to the new version, but there is a need to restart from DOSRES. Before you restart, save the POWER files using POFFLOAD while started from SYSWK1. The restart from DOSRES must be with a POWER COLD start. You can request a COLD start via startup program DTRISTRT.
- VSAM space exhausted during Stage 2. Program IDCONS may be used to define space from the console directly releasing the PAUSEFSU job. IDCONS prompts for IDCAMS commands, the command to define space is as follows:

```
DEFINE SPACE ( TRACKS (150) ORIGIN (12000)  VOLUME(SYSWK1)) -
   CATALOG(VSAM.MASTER.CATALOG)
```

Figure 127. Define VSAM Space During FSU

After space has been defined you may resume the FSU by releasing job DTRFSUAB.

- VTAM does not come up after Stage 2 has finished. This is in most cases due to the fact that not enough DSPACE was defined. You may define more space using SYSDEF AR command SYSDEF DSPACE,DSIZE=20M. The problem might also be caused by an SVA not large enough.
- DTRCLFSU ends with RC=16
  If an external security manager is active, answer *RESUME* to DTRFSUAB. At the PAUSE reply

  ```
  0 // GOTO NOBST
  ```

  The job will finish with all steps for basic start. Then you can proceed with all normal steps for the FSU.
- CICS does not come up after Stage 2 has finished
  For the new release of CICS all CICS tables must be recompiled. A basic start might be helpful.
- IPL from DOSRES does not work after FSU. This is probably due to the fact that the system was configured while IPLed from SYSWK1. All configuration dialogs will write changed procedures to IJSYSR1.SYSLIB on SYSWK1 in this case. The system library however already was copied back to DOSRES before the configuration was made. IPL from SYSWK1 (not with JCL=$$JCLFSU) and copy the changed parts back to IJSYSR2.SYSLIB on DOSRES.

System tries to install PRD1.BASE in step DTRFSU4A into PRD2.PROD.
There is one of the base products installed into PRD2.PROD, for example the high level assembler. Product should be removed first as described in the Planning guide.

At this point, you should cancel DTRFSU4A, release job PAUSEFSU, run MSHP from the console to remove the product (use the work history file). MSHP will prompt you. For example for 3380 this would be for the HLASM:

```
// DLBL IJSYSHF,'WORK.HIST.FILE'
// EXTENT SYS018,SYSWK1,1,0,900,75
// ASSGN SYS018,DISK,VOL=SYSWK1,SHR
// EXEC MSHP
REMOVE 234689
REMOVE 5696-234-00-689
/*
```

Figure 128. Remove a Product from MSHP History File Sample Job

After removing DTRFSU4A may be resumed.

- Problems related to TCP/IP

  In z/VSE 5.2 TCP/IP is installed in library PRD2.TCPIPC instead of PRD1.BASE. This might cause problems if the LIBDEF chain does not contain PRD2.TCPIPC. In this case you can

  – Check, if a wrong job is in the RDR queue , and you can submit the correct job.

  – Check, if you have cataloged a correct job and load it via DTRIINIT.

  – Use the appropriate skeleton in ICCF library 59 to submit a correct job.

- Problems related to external security manager (ESM)
  In a system with external security manager the IPL procedure contains the following statement:

  ```
  SYS ESM=esmphase.
  ```

  Before an FSU is started, the ESM should be upgraded.
  During the FSU, IPL from SYSWK1 takes place. The users own IPL procedure is used, containing the

  ```
  SYS ESM=esmphas
  ```

  statement.
  At this stage the following problems may occur:

  – FSU job DTRCLFSU executes BSTADMIN. This is not possible with the ESM, so the DTRCLFSU job may fail with RC=16.
  You have now 2 possibilities:

    1. proceed with the basic start on SYSWK1 by entering

       ```
       // EXEC PROC=BASICBG
       ```

       Basic VTAM and CICS will be started.
    2. IPL from DOSRES and do the post FSU steps like Update selection panels (PF6) then.

  – The CICS startup may fail with

    ```
    DFHXS1112(indicating the CICS region and groupid could not be determined
    SAF CODES ARE (X'00000004',X'00000000')
    ESM CODES ARE (X'00000000',X'00000000').
    and DFHXS0002 a severe error (CODE X'0109') has occurred in module
    DFHXSDM
    ```

The CICS startup runs per default with SEC=YES. This is controlled by parameter XSECP from CPUVAR1, which has the following default value:

```
// SETPARM XSECP=FB
```

FB is the default partition for the security manager.
Now set XSECP to RECOVER with the following statement

```
// EXEC DTRSETP,SIZE=AUTO,PARM='CPUVAR1;;SET XSECP=RECOVER'
```

With XSECP=RECOVER the CICS will be started with SEC=NO.
When CICS comes up, reset XSECP

```
// EXEC DTRSETP,SIZE=AUTO,PARM='CPUVAR1;;SET XSECP=FB'
```

Replace Bootstrap Record (IJBREPB)

When using program IJBREPB to update/rebuild the bootstrap record, special care should be taken with the required LIBDEF statement. In addition, IJBREPB is restricted to run in a static partition, the current version does not run in a dynamic partition.

Update of the bootstrap record is required when certain changes in the IPL routines are made. The IPL phases for z/VSE are stored in the SVA and addressed via the SDL.

If updates are done which require to rebuild the bootstrap record, the execution of program IJBREPB is required. For the execution of this program, the SDL is searched after the libraries containing the updated IPL phase. With standard LIBDEF processing, the SDL is always searched first. To change this, a LIBDEF PHASE,SEARCH=(lib.sublib,SDL),TEMP is required. As stated in the z/VSE System Control Statements manual, lib.sublib can not refer to IJSYSRS.SYSLIB or IJSYSR1-9.SYSLIB. It is therefore required to have the modified IPL phase in any other z/VSE library (not IJSYSRx) when running IJBREPB, otherwise the original (old) IPL phase will be loaded.

Sample Job for IJBREPB

The following sample job is part of the II service dialogs and copies the IPL phases from IJSYSRS.SYSLIB into a new sublib of library PRD1 before invoking IJBREPB. The job has changed due to SCSI support, do not use an old version of the job to replace the bootstraps.

```
// ASSGN SYS005,DISK,VOL=SYSWK1,SHR
// EXEC LIBR,PARM='MSHP'
DEFINE SUBLIB=PRD1.$$MSHP
CONNECT S=IJSYSR1.SYSLIB:PRD1.$$MSHP
COPY $$A$IPL1.PHASE
COPY $$A$IPL0.PHASE
COPY $$A$PLBF.PHASE
COPY $$A$PLBK.PHASE
COPY $$A$PLBS.PHASE
COPY $IJBFCP.PHASE
COPY $IJBSCSI.PHASE
COPY $IJBLDIPL.PHASE
/*
// ASSGN SYS005,DISK,VOL=SYSWK1,SHR
// LIBDEF PHASE,SEARCH=(PRD1.$$MSHP,SDL)
// EXEC IJBREPB
/*
```

Figure 129. Replace IPL Bootstrap Record Sample Job

Above job takes the new (upgraded) IPL phases from IJSYSR1.SYSLIB. Use skeleton SKBOOTST in ICCF library 59.

Sharing the VSE Control File Between z/VSE Systems

The VSE control file can be shared between z/VSE systems. Only systems with either CICS TS or CICS/VSE can share the control file. It is not possible to share the IESCNTL between VSE/ESA 2.3.x or earlier systems and VSE/ESA 2.4 or later systems.

For sharing, the following steps must be performed:

1. In case of VSE/ESA 2.4 or higher, use *MSG FB,DATA=CLOSECNTL* to close the file in the basic security manager.
2. Close the VSE control file for all CICS TS systems.
   - In case of a pre-VSE/ESA 2.4 system, you only have to close the file in all CICS systems using *CEMT SET FILE(IESCNTL) CLO*
3. Alter the shareoption to (4 4) using IDCAMS
4. In case of VSE/ESA 2.4 or higher, open the VSE control file from the basic security manager using *MSG FB,DATA=OPENCNTL*
5. Open the VSE control file.
   - In case of a pre-VSE/ESA 2.4 system, use *CEMT SET FILE(IESCNTL) OPEN* for all CICS systems.

```
// JOB SHARE CHANGE SHAREOPTION
* MAKE SURE THAT THE CONTROL FILE IS CLOSED
// EXEC DTRIATTN,PARM='MSG FB,DATA=CLOSECNTL'
// EXEC DTRIATTN,PARM='MSG F2,DATA=CEMT SET FILE(IESCNTL) CLO'
/*
// EXEC IDCAMS
   ALTER VSE.CONTROL.FILE.@D@ -
   SHAREOPTIONS(4 4)                  -
   CATALOG(VSESP.USER.CATALOG)
   /**/
   ALTER VSE.CONTROL.FILE.@I@ -
   SHAREOPTIONS(4 4)                  -
   CATALOG(VSESP.USER.CATALOG)
   /**/
/*
* MAKE SURE THAT THE CONTROL FILE IS OPENED
// EXEC DTRIATTN,PARM='MSG FB,DATA=OPENCNTL'
// EXEC DTRIATTN,PARM='MSG F2,DATA=CEMT SET FILE(IESCNTL) OPE'
/*
/&
```

Figure 130. Sharing the Control File

Move the VSE/POWER Data File

The VSE/POWER data file is located on SYSWK1.

You may want to move the Power Data File to an other volume for performance reasons.

To do so, the following steps must be performed.

1. Change the label procedure STDLABEL in IJSYSRS.SYSLIB
2. Copy the changed procedure to PRD2.SAVE (needed for FSU).
3. Change the assignment procedure DTRPOWR in IJSYSRS.SYSLIB
4. Copy the changed procedure to PRD2.SAVE (needed for FSU).
5. Request a VSE/POWER cold start using DTRSETP
6. Shut down the system except VSE/POWER
7. Backup the VSE/POWER queues.
8. IPL the system, VSE/POWER cold start is performed
9. Restore the VSE/POWER queues

The skeleton SKPWREXT in ICCF library 59 describes the steps in detail.

Note: Refer also to the Data File extension at warm start function in *VSE/POWER Administration and Operation*.

Extend the VSE Dump Library

The VSE Dump Library holds dumps from all static and dynamic partitions. If the dump library is full, dumps are written directly to SYSLST.

Starting with z/VSE 5.1 the dump library is located in VSAM space. The initial installation job VSAMDEFS defines the SYSDUMP library in the master catalog.

	FBA	**ECKD**
PRIMARY	81920	1800
SECONDARY	81920	1800

Having the VSE dump library in VSAM space has a big advantage. If it is running out of space, VSAM automatically extends the file. If needed, only the space for the master catalog must be extended.

If your dump library is in BAM space and you need to enlarge the VSE dump library use skeleton SKDMPEXT in ICCF library 59 to do so.

The steps below must be performed:

Be aware that using the dump archive PRD2.DUMP can give relief of this problem, since the dump archive is located in VSAM space which extends automatically.

1. Delete the dumps, which are not needed.
2. Save the dumps, which are still needed (offload to tape).
3. Change the label procedure STDLABEL in IJSYSRS.SYSLIB
4. Copy the changed procedure to PRD2.SAVE (needed for FSU).
5. Since the dump sublibraries are assigned to the corresponding partitions via *LIBDEF DUMP* they can not be deleted directly, but only in a system without assignments to the dump sublibraries.
6. Shut down your system
7. IPL with a *mini start*. Note that the mini start does not have assignments to the dump sublibraries.
8. Define the SYSDUMP library and the sublibraries. The member LIBRDEFS.Z in IJSYSRS.SYSLIB may serve as example how to do this.
9. IPL your normal system.
10. Run the *DUMPINIT* job to initialize the dump library. You may use *SKDMPINI* in ICCF library 59 to do it.
11. You may onload necessary dumps from tape.

SMF-type Job Accounting

A job account exit is provided as a skeleton which creates CICS SMF type records. Thus various reports can be generated out of the account data. The following describes how to implement SMF Job Accounting.

1. Copy the following skeletons from ICCF library 59 to your private library:
   - SKJADACC
   - SKJADOFF
   - SKJOBDMF
   - SKJADPRT
2. Submit SKJADACC for cataloging the new job accounting routine and defining the account dataset. The job will replace any existing Job Account exit. There is a PAUSE statement coded before the exit is replaced. After running the skeleton, Job Accounting starts recording, all following job steps and jobs will be analyzed. In addition, utility SMFCLEAN is established.
3. After the accounting is active, submit SKJADOFF to offload the account data to a working dataset. This job will offload from SMF.ACCT.FILE. to the ACCTREPP.WORK file. It also moves the contents of the accounting buffers to the file using utility SMFCLEAN.
4. Submit SKJOBDMF. This is the phase for JA reporting and generates the phase ACCTREP.
5. Edit SKJADPRT and adapt the following:
   - // ASSGN SYS010,02E to a printer address in your installation
   - * $$ LST,02E to a printer address in your installation
   - 'PARM=.....' Statement to your needs. E. G. PARM='JSTEPO' prints all job steps.

The following are parameters available for reporting selection with SKJADPRT:

```
*           "JSTEPO"
*           (JOB STEP REPORT ONLY)
*                           (DEFAULT NOTHING)
*
*           "JSTEP"
*           (JOB STEP REPORT AND DEFAULTS FOR JOB SUMMARY ACTIVE
*                           (DEFAULT NOTHING)
*
*           "STA=PARTID1,PARTID2,...PARTID12,END"
*           (PARTID=BG F1 F2 ..)
*           EXAMPLE:  STA=F1,BG,F3,END   SPECIFIC STATIC PARTITONS
*                                         MAX : 12 PARTITIONS
*                             (DEFAULT ALL STATIC PARTITONS)
*
*           "DYN=CLASS1,CLASS2,SPEZCLASS3,SPEZCLASS4,END"
*           (CLASS1= C B Z ..)  (SPEZCLASS1=W1 V3 Z4 ..)
*           EXAMPLE:  DYN=C,B,G,Q1,S3,END  ALL JOBS FOR A DYN CLASS
*                              OR ONLY FOR A SPECIFIC DYN CLASS
*                                         MAX : 25 DYN INPUTS
*                               (DEFAULT ALL DYNAMIC PARTITIONS)
*
*           "RANGE=DDMMYYYY-DDMMYYYY"
```

```
*           (RANGE= TIME FRAME BETWEEN WANTED JOBINFO COLLECTION)
*           EXAMPLE:  RANGE=11041999-14051999
*                                 (ALL JOBINFO)
*
*           "SORT=JNAM‘JDAT"
*           (JNAM= SORTED BY JOBNAMES JDATE= SORTED BY JOBDATE)
*           EXAMPLE:  SORT=JNAM
*                                 (DEFAULT SORTED BY DATE )
*
*           "RECNUM=X"
*           (X= 1 TO N ) X= VALUES OF MINIMUN AMOUNT OF AVAILABLE
*                          JOB INFO FOR COLLECTING
*           EXAMPLE:  RECNUM=5
*           AT LEAST JOBS WITH 5 INFOS  OF THE SAME JOBNAME WILL BE
*           COLLECTED
*                                 (DEFAULT 10 )
*
*           "SKIP=X"
*           (X= 1 TO 49) X= VALUE IN % TO CUT AT THE BEGINING AND
*           AT THE OF JOBCOLLECTION
*           EXAMPLE:  SKIP=15
*           THIS MEANS 15% WILL BE REMOVED AT THE BEGINNING AND AT
*           THE END OF EVERY JOBNAME COLLECTION
*                                 (DEFAULT 10 )
*
*           "EXCHN=A,B-C,D,END"
*           (A = REPRESENT A CHANNEL VALUE BETWEEN 0-F)
*           RANGE ARE ALSO ALLOWED E.G. 3-4
*           EXCLUDE ALL DEVICES AN CHANNEL BETWEEN 3 AND 4
*           DEFAULT : NO CHANNEL EXCLUSION
*
*
*           "EXDEV=CAA,CBB-CCC,CDD"
*           (CAA REPRESENT A DEVICE ADDRESS)
*           RANGE ARE ALSO ALLOWED E.G. 220-250
*           EXCLUDE ALL DEVICES ADDR 220 TO 250
*           DEFAULT : NO DEVICE EXCLUSION
```

REXX program DMPMGR to Manage the Dump Library

A REXX procedure is provided that takes precautions against a full dump library. The program can handle the dump library SYSDUMP and the dump archive PRD2.DUMP. The program determines the current size of the dump library or dump archive. If a certain limit is exceeded, certain actions are taken:

- a console message is written
- dumps of certain partitions are deleted
- dumps of certain partitions are printed
- dumps of certain partitions are offloaded

A date can be specified to handle dumps only that are created during this date or earlier. Or an age can be specified to handle dumps only that are older than the given age.

Printing, and offloading of dumps are done by INFOANA invocations in separate POWER jobs. Deletion of dumps is done within one POWER job.

The parameters of REXX program DMPMGR are:

LIB=lib or lib.sublib
name of the library where the dumps reside. This parameter is optional. Default is SYSDUMP.

MSG

LIMIT=nn percentage of used library space that must be exceeded to trigger actions. Default is 90%. LIMIT is ignored for PRD2.DUMP.

MSG issue the highlighted console message "!!! Dump Library is almost full !!!", if dump library runs full. This parameter is optional.

DELETE=(p1,...,pn) or DELETE=p1
list of partitions whose dumps are to be deleted if dump library runs full. This parameter is optional.

PRINT=(p1,...,pn) or PRINT=p1
list of partitions whose dumps are to be printed if dump library runs full. This parameter is optional.

OFFLOAD=(p1,...,pn) or OFFLOAD=p1
list of partitions whose dumps are to be saved on tape. This parameter is optional.

TAPE_UNIT=ttt
tape unit to be used for offloading dumps

DATE=yy-mm-dd
date limit for dump processing. This parameter is optional. Default is the current date.

AGE=n age limit in days for dump processing. This parameter is optional. Default is 0 days.

CLASS=p
VSE/POWER job class of generated jobs. This parameter is optional. Default is CLASS Y.

DISP=D or DISP=H
disposition of generated jobs. This parameter is optional. Default is DISP=H.

Here are 2 invocation samples: (see also SKDMPMGR in ICCF-lib 59)

```
// EXEC REXX=DMPMGR,PARM='LIMIT=85 DELETE=BG MSG PRINT=F2'
```

when using the PARM-operand to specify parameters.

```
// EXEC REXX=DMPMGR
MSG
DELETE=(F5,F6,F7)
OFFLOAD=(F2,F8)
LIMIT=89
TAPE_UNIT=181
DATE=03-04-30
CLASS=C
DISP=D
```

when using SYSIPT to specify parameters.

DMPMGR ends with one of these Return Codes:

0	successful, limit not exceeded
1	successful, limit exceeded
4	syntax error
8	invocation of LIBR LD L=SYSDUMP failed
12	submission of INFOANA - PRINT DATA job failed
14	retrieval of INFOANA - PRINT DATA job output failed
16	submission of INFOANA - PRINT job failed
18	submission of INFOANA - OFFLOAD job failed
20	submission of INFOANA - DELETE job failed

The following skeleton SKDMPMGR in ICCF library 59 demonstrates how to schedule such a job to handle the dump file repetitively. The job below with the given operands is scheduled every 30 minutes every day.

```
* $$ JOB JNM=DMPMGR,CLASS=Y,DISP=K,
* $$ DUETIME=0000,DUEDAY=DAILY,DUEFRQ=(0030,2400)
// JOB DMPMGR - REXX DUMP MANAGER
* ****************************************************************
*
*  ------ INVOKE REXX DUMP MANAGEMENT PROCEDURE --------------------
*
*  THIS SKELETON MAY BE USED TO CHECK CURRENT USAGE OF DUMP SPACE.
*  IF FILLED TO A CERTAIN DEGREE, CERTAIN ACTIONS ARE INITIATED.
*
* ****************************************************************
// EXEC REXX=DMPMGR,PARM='LIMIT=85 TAPE_UNIT=181 MSG'
OFFLOAD=(F7 F8)
PRINT=(BG,DYN)
DELETE=F2
DISP=D
/*
/&
* $$ EOJ
```

Figure 131. Periodic Dump Library Management Job Example

REXX program REXDFHDU to print CICS Dumps selectively

A REXX procedure is provided that can be used to print CICS DUMP selectively. This program performs the following steps:

- determines current CICS dump dataset and its status (Opened or Closed)
- if necessary, asks whether to switch or close the active CICS dump dataset and does switching or closing
- sets the label DFHDUMP for the desired dump dataset
- scans the CICS dump dataset with DFHDU420 (TYPE=SCAN)
- determines dumps to be printed (console dialog)
- prints selected dumps with DFHDU420
- if necessary, reopens the dump dataset

The parameters of REXX program REXDFHDU are:

CICS={CICSICCF|CICS2|other }
specifies the desired CICS, default: CICSICCF

DUMP={A|B }
specifies the desired dump dataset (A or B), default: dump dataset currently active in CICS

LABEL={Extern|Intern }
specifies whether the label for DFHDUMP is set outside (Extern) or inside (Intern) the REXX proc.
default: Intern for CICSICCF and CICS2
Extern for other CICS

Here are 2 invocation samples: (see also SKDMPMGR in ICCF-lib 59)

```
// EXEC REXX=REXDFHDU,PARM='CICS=CICSICCF LABEL=INTERN'
```

when using the PARM-operand to specify parameters.

```
// EXEC REXX=REXDFHDU
CICS=CICSOLD
DUMP=B
LABEL=EXTERN
```

when using SYSIPT to specify parameters.

Progaram REXDFHDU ends with one of these Return Codes:

0 successful

8 unsuccessful, see console messages

The following Job sample demonstrates how to activate REXX Program REXDFHDU to print a CICS dump using external labels of non default CICS (CICSPRO1) for DFHDUMP.

```
* $$ JOB JNM=REXDFHDU,CLASS=0,DISP=D
* $$ LST CLASS=V,DISP=D,DEST=(*,REXDFHDU)
// JOB REXDFHDU    SCAN AND PRINT CICS TRANSACTION DUMPS
* ****************************************************************
*
*  ------ INVOKE REXX  PRINT CICS DUMP PROCEDURE -------------------
*
* ****************************************************************
// DLBL DFHDUMP,'CICSPRO1.DUMPA',0,VSAM,CAT=CICSUSR,DISP=(OLD,KEEP)
// LIBDEF *,SEARCH=(PRD1.BASE)
// LIBDEF PROC,SEARCH=(PRIMARY.HSCZ,PRD1.BASE)
// EXEC REXX=REXDFHDU,PARM='CICS=CICSPRO1  LABEL=EXTERN'
/*
/&
* $$ EOJ
```

Figure 132. Print CICS dumps selectively Job Example

DTRIATTN Utility to Execute Console Commands

The DTRIATTN utility allows to execute console commands in a batch job. The commands to be executed are passed to the system using the SVC 30 interface. There are some limitations:

- Commands must be passed on to the program in the PARM field, only one PARM field is allowed.
- The length of the parameter is limited to 100 characters, however the SVC 30 routine has a limit of 72 characters.
- Only one command can be executed at a time.

Following are samples:

- POWER command to delete the accounting file:

```
// EXEC DTRIATTN,PARM='J DEL'
/*
```

- Closing of files open in CICS:

```
// EXEC DTRIATTN,PARM='MSG F2,DATA=CEMT SE FI(INWFILE) CLOSE'
/*
```

- You can use STACK to issue multiple commands:

```
// EXEC DTRIATTN,PARM='STACK LIBR1|R RDR PAUSEBG|0 EXEC LIBR|'
/*
// EXEC DTRIATTN,PARM='STACK LIBR2|0 AC S=IJSYSRS.SYSLIB|0 END|0|'
/*
// EXEC DTRIATTN,PARM='LIBR1'
/*
// EXEC DTRIATTN,PARM='LIBR2'
/*
```

IESVCLUP Utility to Add VSAM Labels to STDLABUP Procedure

The IESVCLUP utility allows to add or delete VSAM standard labels to or from a label procedure. The label procedure has to reside on IJSYSRS.SYSLIB. The utility reads input from SYSIPT, the input is an 80 character input card with positional parameters as follows:

Column 1
One character function, A = add, D = delete

Column 3-46
44 character file id

Column 48-54
7 character file name

Column 56-62
7 character catalog name

Column 63-66
3 character disposition allowed values are OLD and NEW

Column 68-71
4 character disposition allowed values are KEEP, DEL or DATE

Column 72-80
8 procedure name, default name is STDLABUP

A sample adding a label for a library would be:

```
// EXEC IESVCLUP,SIZE=AUTO
A VSE.AMADLIB.LIBRARY                          AMADLIB VSESPUC OLD KEEP STDLABUP
/*
```

A sample deleting a label would be:

```
// EXEC IESVCLUP,SIZE=AUTO
D                                              INWFILE
/*
```

DFHCSDUP Utility

The DFHCSDUP utility can be used to list or change definitions in the CSD file.

Be aware that the size of the DFHCSDUP program has changed with CICS TS for z/VSE 2.2.
For the call of DFHCSDUP we recommend to use the the following form:

```
// EXEC DFHCSDUP,SIZE=DFHCSDUP
```

In case you use skeleton SKLE370 check and update the call of DFHCSDUP.
If you use the Debug Tool check and update EQATCSD and EQACCSD.

Application of Service

The Interactive Interface offers two ways to apply service,

1. using mass application. This is fastpath 1423 with TYPE=ALL specified. The system will apply all PTFs on tape/disk as long as they are not yet installed on the system, they are not superseded by PTFs on the same tape or disk and as long as all requirements for the PTF are fulfilled. This process also supports multiple tapes.
2. selective application. This is dialog Analyze and Apply PTFs, fastpath 1422. This dialog is preparing an overview of the PTFs on the service tape/disk. Especially if installing many PTFs like for a PSP bucket or an RSL, this can be very difficult and complex. In addition, the more PTFs, the more the dialog needs storage. Since the storage is needed in the ICCF pseudo partition class I, it might help to get the dialog to work stopping ICCF and restarting it with a larger partition layout. to do so, enter following commands on the console:

   ```
   /ICCFEND
   ```

   When ICCF is down restart it with:

   ```
   MSG F2,DATA='I$ST DTSIGENM'
   ```

In general, the mass application is recommended. Concerning direct or indirect application, forcing indirect application in the dialog allows to first test the applied service, but this requires an ipl from SYSWK1. If direct application is selected, service will be applied indirect - also requiring an ipl from SYSWK1, if at least one PTF was marked for indirect application. Such PTFs are usually for advanced functions e.g. affecting the supervisor.

Concerning installation of service delivered electronically, refer to the System Upgrade and Service manual or to the VSE home page:

`http://www.ibm.com/systems/z/os/zvse/documentation/edelivery.html#eptf`

Application of Service from Disk

The Interactive Interface offers the possibility to apply service from disk. For this task the file IJSYSPF PTF.FILE is defined in the system.

In z/VSE 5.2. the IJSYSPF file was delivered with an incorrect definition. In order to apply service from disk it is necessary to redefine this file prior to service application.

This can be done by changing and then submitting skeleton SKPTFILE in library 59 from:

```
RECORDSIZE (80,10320)                -
RECORDFORMAT(FIXBLK(10320))          -
```

to

```
RECORDSIZE (80,10320)                -
RECORDFORMAT(FIXBLK(80))             -
```

The skeleton SKPTFFILE is corrected in PTF UI17100.

Once the IJSYSPF file has been redefined, if a PTF is to be transferred using TCP/IP the file must be transferred as follows:

```
ftp> binary <-- switch to binary mode
200 Command okay.
ftp> quote site recfm fb <-- record format of your file
200 Command okay.
ftp> quote site lrecl 80 <-- record size of your file
200 Command okay.
ftp> quote site blksize 10320 <-- blocksize of your file
200 Command okay.
ftp> put ptffile.bin PTF.FILE <-- enter your filenames
```

Problem during PTF application of a HLASM PTF

When you create a job to apply High Level Assembler (HLASM) PTFs containing I-books, the job step DTRPTF05 may fail with:

```
0S03I PROGRAM CHECK INTERRUPTION  - HEX LOCATION 09C86674 -
INTERRUPTION CODE 04 - PROTECTION EXCEPTION
0S00I JOB DTRPTF05 CANCELED
```

In DTRPTF05 the HLASM is used to catalog I-book members into ICCF library. If HLASM phases are in the SVA, the usage of HLASM prior to reloading the phases into the SVA may suffer from inconsistencies.

To solve this problem perform the following steps.

1. Reply EXIT to DTRPTFAB to exit PTF application.
2. Resolve the inconsistency by
   a. Reload the HLASM phases into the SVA:

      ```
      Run in BG
      // SET SDL
      LIST=$SVAASMA
      /*
      ```

b. Or IPL the system

3. Release job DTRPTFAB
4. Answer RESUME to start the job DTRPTF05 again.

Such a critical PTF is e.g UI25027.

Apply PTF from the internet

To apply a PTF from the internet is described at the z/VSE homepage at http://www-03.ibm.com/systems/z/os/zvse/documentation/edelivery.html#ept

Using the Host Transfer File (HTF)

For PTF application first the PTF is moved to PTF.FILE.
When the host transfer file is used for this step, a problem will occur.
The dialog 383 will display the message

```
MOVE COMPLETED; ONE OR MORE RECORDS WERE NOT MOVED, SOURCE FILE NOT
DELETED
```

. The following solutions are possible:

- Use a library member instead of HTF, for example PTFFILE.Z in library PRIMARY.SUF.
  The following send command will transfer the PTF to the PRIMARY library:

  ```
  SEND PTFFILE.BIN A: PTFFILE Z (FILE=LIB L=PRIMARY S=SUF LRECL=80 BINARY
  ```

  Transfer the PTF file to disk with the following DITTO job:

  ```
  * $$ JOB JNM=COPYPTF,CLASS=0,DISP=D
  // JOB COPYPTF
  * PTF MEMBER IS COPIED (AND REBLOCKED)
  * TO VSAM FILE IJSYSPF
  // LIBDEF *,SEARCH=(PRIMARY.SUF)
  // UPSI 1
  // EXEC DITTO
  $$DITTO LV LIBIN=PRIMARY.SUF,
  $$DITTO MEMBERIN=PTFFILE.Z,
  $$DITTO FILEOUT=IJSYSPF,
  $$DITTO REUSE=YES
  /*
  /&
  * $$ EOJ
  ```

- Use the HTF with changed definition of the PTF.FILE
  The dialog Move Utility (IUI fast path: 383) can not handle the PTF.FILE with the properties from the actual definition.
  If you use the HTF, you have to delete and define the PTF.FILE with ed parameters:
  In SKPTFFILE replace

  ```
            RECORDSIZE (80,10320)                 -
  by
            RECORDSIZE (80,80)                    -
  ```

  Following send command will transfer the PTF to the HTF (IUI fast path 386):

  ```
  SEND PTFFILE.BIN A:PTFFILE Z (FILE=HTF LRECL=80 BINARY
  ```

From the HTF the PTF file can be transferred to disk. The Move Utility (IUI fast path: 383) from HTF to VSE/VSAM can be used. It is required that the IJSYSPF file is defined to CICS. After the move to VSE/VSAM, IJSYSPF needs to be closed

```
CEMT SET FILE(IJSYSPF) CLOSE
```

.

Note: If you change the PTF.FILE definition take care whenever use a different way of data transfer later; the values described would not fit. Best is to delete the PTF.FILE after PTF application.

Now the PTF.FILE contains the PTF, you can apply service using the dialog, service medium is disk.

Transferring Tape Image Files to virtual Tape

z/VSE 6.2 is delivered on DVD or through electronic delivery via Shopz. The z/VSE base tape, the z/VSE extended base tape or optional product tapes are available in AWS format, namely VSE620EN.AWS, VSE620XB.AWS or VSE620OP.AWS.

The installation of products from the AWS file is not possible, but you need a procedure, to mount the AWS file as a virtual tape.

We recommend to use TCP/IP.
You can access the file as remote virtual tape (see skeleton SKVTACPY) or you can FTP the AWS file to a VSAM data set.

Without TCP/IP, you can transfer the AWS file using the workstation file transfer into the Host Transfer File (HTF) and from there to a VSAM data set.

Use workstation file transfer via the host transfer file

The terminal must be in DFT mode. See **"Basic Problem Solving Questions" on page 269** in chapter **Chapter 17, "IWS File Transfer" on page 269**.
Please do the following steps:

1. Define the host transfer file
   - Use skeleton SKIWSTF from ICCF library 59
   - increase the number of records from
     RECORDS (4000 4000) - to RECORDS (4000 50000) -
   - make sure that there is enough space in the VSE user catalog VSESPUC to hold the host transfer file and the virtual tape file VTAPE1.
2. Define the virtual tape file VTAPE1 in VSAM
   - Use skeleton SKVTAPE from ICCF library 59
3. Open the host transfer file and the virtual tape file
   - CEMT SET FILE(INWFILE) OPEN
   - CEMT SET FILE(VTAPE1) OPEN
4. Transfer the AWS tape image to the host transfer file

- In an IUI terminal session (for example session A) either hit PF6/PF9 or select PC File Transfer (fastpath 386). The userid will be the owner of the file.
- On the workstation command line issue for IBM Personal Communication

  ```
  SEND  VSE620XB.AWS a: VTAPE1 AWS (FILE=HTF BINARY NOCRLF
  ```

  where *a* indicates the session.

5. Transfer the file from the host transfer file to the VSAM dataset (via dialog or via JCL)
   - via dialog
     - Select Move Files from the Host Transfer File to VSAM (fastpath 383) or select *List and Process User Files in Host Transfer File* (fastpath 381) and option 7 (move to VSAM)
   - via JCL
     - Close the host transfer file

       ```
       CEMT SET FILE(INWFILE) CLOSE
       ```

     - In case you have defined the virtual tape file VTAPE1 in CICS, make sure that it is closed.

       ```
       CEMT SET FILE(VTAPE1) CLOSE
       ```

     - Use the following JCL

       ```
       // DLBL VTAPE1,'VSE.VTAPE.FILE',99/366,VSAM,CAT=VSESPUC
       // EXEC INWMUTIL,SIZE=AUTO
       UNLOAD,FILENAME=VTAPE1,FILETYPE=AWS,USERID=owner-of-file
       /*
       ```

       Note that the DLBL file name allows only 7 characters, which must match the first 7 characters of the file name from the HTF.

Refer to *z/VSE System Utilities* and *z/VSE Programming and Workstation*.

Transferring Raw Dumps to and from VSE Dump Library

VSE dumps can be transferred in binary format to a PC and back to VSE using workstation file transfer or TCP/IP based FTP. The raw format dump is used for example to format a CICS dump on VSE using the Interactive Interface. VSE has its own naming convention concerning dumps. In order to format a dump on VSE, this naming conventions should be observed, dump names are of the form Dxxyyyyy where xx is the partition e.g. F1 and yyyyy is a number. The system starts generating dumps with number 00001, already generated number are stored in the dump library. Transferring a dump to the VSE dump library can be done replacing any existing dump or using a new dump name. Dumps in the dump library have to have a file type of DUMP.

Following is a sample to transfer a dump into the dump library, transferring it to the PC and also transferring it to VM so that it can be used by the VSEDUMP utility. Following samples are using workstation file transfer, emulator session A is assumed. Transfer a dump stored on PC named mydump.dmp to a dump named DF200001 into SYSDUMP.F2 for later formatting with CICS TS:

```
send mydump.dmp a:df200001 dump (file=lib l=sysdump s=f2 binary
```

Receive a dump out of the dump library to the PC:

```
receive mydump.dmp a:df200001 dump (file=lib l=sysdump s=f2 binary
```

The raw dump on the PC can be packed e.g. using PKZIP for transferring i to IBM. File transfer to VM suitable for VSEDUMP is done as

```
send mydump.name a:mydump vsedump t (lrecl 4112
```

Using TCP/IP FTP works similar way. First you have to define the dump library to TCP/IP as follows:

```
DEFINE FILE,TYPE=LIBRARY,DLBL=SYSDUMP,PUBLIC='SYSDUMP',ALLOWSITE=NO
```

With this definition, a file transfer using ftp would look like:

```
C:Ö>ftp 9.164.155.2              <---------- your IP address
Connected to 9.164.155.2.
220-TCP/IP for VSE -- Version 01.04.00 -- FTP Daemon
    Copyright (c) 1995,2001 Connectivity Systems Incorporated
220 Service ready for new user.
User (9.164.155.2:(none)): sysa   <------------- your user id
331 User name okay, need password.
Password:                         <--------------- your password
200 Command okay.
ftp> cd sysdump
250 Requested file action okay, completed.
ftp> cd f2                        <---- Sublib where your dump resides
250 Requested file action okay, completed.
ftp> bin                          <-- switch to binary mode
200 Command okay.
ftp> get DF200001.dump            <-------------- Name of the dump
200 Command okay.
150-About to open data connection
    File: SYSDUMP.F8.DF800004.DUMP
    Type: Binary Recfm: S  Lrecl:  4096
    CC=ON  UNIX=OFF RECLF=OFF TRCC=OFF CRLF=ON
150 File status okay; about to open data connection
226-Bytes sent: 23,273,920
    Records sent:           711
    Transfer Seconds:         6.91 (  3,788K/Sec)
    File I/O Seconds:         5.30 (  4,545K/Sec)
226 Closing data connection.
ftp: 23273920 bytes received in 7,51Seconds 3098,64Kbytes/sec.
```

The way back is done using PUT instead of GET. Transferring to VM, make sure the logical record length is set to 4112, to do so you have to specify:

```
quote site lrecl 4112
```

before you issue the PUT command.

If there is a need to transfer the dump from VM back to VSE, the transfer from VM is a normal binary transfer. This transfer from VM to VSE can also be done directly without involving a workstation.

For detailed information and instructions see
www.ibm.com/systems/z/os/zvse/support/problem.html

Handling of large dumps

Large dumps (that is, dumps of partitions that are larger than approximately one GB) cannot be uploaded from the dump tape into z/VSE using the INFOANA ONLOAD utility. This is because for such large dumps, the INFOANA utility cannot obtain enough storage below the 16 MB line. The job will end with these return codes:

```
BLN9002I ERROR IN EXTERNAL ROUTINE, RETURN CODE = 12, REASON CODE = 712
BLN3002I ONLOAD FAILED, REASON CODE = 3012
```

To avoid this problem, you can use a DITTO job as listed below to upload the dump file into the z/VSE dump library. In this job, file number 6 represents the partition dump.

```
// EXEC DITTO
$$DITTO REW OUTPUT=181
$$DITTO FSF OUTPUT=181,NFILES=5
$$DITTO TL INPUT=181,RECFMIN=FB,LIBOUT=SYSDUMP.BG,
$$DITTO    MEMBEROUT=DBG00001.DUMP,RECFMOUT=S
/*
```

If the dump was taken on a z/VM system, you can upload the dump file into the z/VM dump library using these CMS statements:

```
TAPE REW
TAPE FSF 5
FILEDEF OUT DISK dump_name type mode ( RECFM F LRECL 4112 BLKSIZE 4122
FILEDEF IN TAP1 (RECFM F LRECL 4112 BLKSIZE 4112
MOVEFILE IN OUT
```

Note: In the above statements, File 6 (forward tape file 5) is the partition dump on the tape. If required, you can now transfer the dump from the z/VSE or z/VM dump library to IBM Support (for further analysis).

If the dump is about 2 GB in size, the librarian can not handle the dump. You get message

```
DIT5500i Library request for PUT failed, RC=16,RSCD=74
```

Transaction Security

In the transaction security dialog you may specify the CICS region.

The CICS region is the userid specified in the ID statement in the CICS startup job (without password).

This userid must be defined in the VSE Control File.

In the predefined z/VSE system it is the same as the APPLID of this CICS in the SIT.

z/VSE provides two of such userids, DBDCCICS and PRODCICS.

If you change your CICS region or if you want to use a new one, make sure that you have defined the corresponding userid in the Control File (use DBDCCICS as model).

To use the CICS region prefix for transaction security makes only sense if the security prefix is specified as on (SECPRFX = YES) in the SIT. (DFHSITSP in ICCF library 59)

Prefix checking works the same way with the newly introduced security concept coming with 3.1.1.

How to define and use auxiliary trace file B

CICS offers the possibility to make an auxiliary trace. Auxiliary trace entries are directed to one of two auxiliary trace datasets, dataset A (DFHAUXT) and B (DFHBUXT).
In z/VSE the auxiliary trace dataset A is ready to be used.
The standard labels for VSAM files STDLABUP.PROC contain a label for DFHAUXT, which allows implicit definition as a SAM/ESDS file in the VSAM user catalog, that means, the file is automatically created at the first usage.

```
// DLBL DFHAUXT,'CICS.AUXTRACE',0,VSAM,                              X
               CAT=VSESPUC,RECSIZE=4096,                             X
               DISP=(NEW,KEEP),RECORDS=(400,0)
```

If you wish to use the auxiliary trace dataset B (DFHBUXT), you have to do the preparation yourself.

The following skeletons are provided in ICCF library 59:

DFHAUXB define auxiliary trace dataset B

DFHAUXPR print trace dataset A

Skeleton DFHAUXB used for DBDCCICS

A label for DFHBUXT must be added. The parameters RECSIZE and RECORDS ensure, that implicit definition can take place.

After submitting DFHAUXB an additional step is necessary.

Add the following lines to STDLABUP.PROC in IJSYSRS.SYSLIB and PRD2.SAVE.

```
// DLBL DFHBUXT,'CICS.BUXTRACE',0,VSAM,                              X
               CAT=VSESPUC,DISP=(NEW,KEEP),                          X
               RECORDS=(400,0),RECSIZE=4096
```

Skeleton DFHAUXPR

You can use DFHAUXPR as a sample to print the auxiliary trace dataset B.
To do so, simply change the DLBL statement as follows:

```
// DLBL DFHAUXT,'CICS.BUXTRACE',0,VSAM,                              X
               CAT=VSESPUC,RECSIZE=4096,                             X
               DISP=(OLD,DELETE),RECORDS=(400,0)
```

Skeletons using the installation tape

The following skeletons are provided to restore data from the z/VSE installation tape and thus repairing destroyed data.

SKTCPRST	Restore TCP/IP or IPv6 from the installation tape
SKOMERST	Restore the online message file
SKICFRST	Selectively restore the DTSFILE
SKREFRE	Restore DTRIHIST.Z from the installation tape
SKRSTRFL	Restore IESTRFL.V from the installation tape

Figure 133. Skeletons using the installation tape

Chapter 19. IPv6/VSE

IPv6/VSE

Starting with z/VSE 6.1, IPv6/VSE V1.2 is located on the z/VSE base tape and is automatically installed in library PRD2.TCPIPB during initital installation.

Be aware that

- During initial installation the configuration dialog TCPCONF allows the configuration of TCP/IP for z/VSE only.
- IPv6/VSE must be configured using provided skeletons.

Skeletons supporting IPv6/VSE

In ICCF library 59 the following skeletons are provided.

SKIPV4ST	Startup job for IPv6/VSE IPv4 stack
SKIPV6ST	Startup job for IPv6/VSE IPv6 stack

Figure 134. Startup skeletons for IPv6/VSE

If you use IPv6/VSE instead of TCP/IP you have to examine and adapt the following jobs / skeletons.

SKCICS	Startup job for CICS/ICCF
SKCICS2	Startup job forsecond CICS
SKLIBCHN	Skeleton for LIBDEF procedure
SKSTGDPS	Startup for the GDPS client
SKSTMAS	Startup for the SNMP monitoring agent
SKSTTRAP	SNMT version 1 trap
SKVCSSTJ	Startup job for VSE connector server
SKVTASTJ	Startup job for virtual tape server

Figure 135. Skeletons to adapt for IPv6/VSE

Chapter 20. VSE e-business Connectors

Overview

The z/VSE e-business Connectors consist of several components:

- The Java-based Connector, which consists of the z/VSE Connector Server (part of VSE/ESA 2.5 and higher) and the z/VSE Connector Client (W-book IESINCON.W in PRD1.BASE/PRD2.PROD, also downloadable from the z/VSE Connectors Internet page (see “More Information and FAQs” on page 327) The z/VSE Connector Client consists of a Java class library, providing access to z/VSE resources from any kind of Java program (applets, servlets, EJBs, etc.), and extensive online documentation and coding samples.
- The VSE Script Server provides access to VSE resources using the Java based connectors, but without writing Java programs. It is used to execute so called VSE scripts. VSE scripts are written in a simple script language. It provides statements like if, while, for to control the program flow. VSE scripts can use various script commands to access VSE resources. A VSE Script can be invoked by any kind of programs (especially non Java Programs). The downloadable package contains samples to invoke a VSE script from office applications like Microsoft Excel or Lotus 1-2-3. The samples shows how to read a VSAM record and include it into a spread sheet.
- The VSE/VSAM Redirector Connector, which allows to redirect access to a particular VSAM file to any remote platform, for example into a DB2 database on a Linux on zSeries. For download and online information refer to the web page “More Information and FAQs” on page 327.

Note: the z/VSE Navigator Function is not part of the z/VSE Connector component. It is a sample that shows the functionality of the Java-based Connector. The z/VSE Navigator is not shipped as part of VSE, but it can be downloaded freely from the z/VSE Connectors and Utilities web page (see “More Information and FAQs” on page 327)

If you have problems with the z/VSE Navigator, send an e-mail to

- zvse@de.ibm.com

Known Problems

The following details information to aid the user in problem solving.

Installation of the z/VSE Connector Client

For a detailed description of how to install the z/VSE Connector Client on your workstation, please refer to the z/VSE Connectors and Utilities web page, see below.

There you will find also information about downloading Java Development Kits (JDKs) and Java Runtime Environments (JREs), which are prerequisite for working with the Java-based Connector.

IESC1017E SYNTAX ERROR IN CONFIG FILE

The messages

```
IESC1017E  SYNTAX ERROR IN CONFIG FILE: member-name
```

(and also)

```
IESC1016E  SYNTAX ERROR IN PLUGIN CONFIG FILE: member-name
```

normally occur when the z/VSE Connector Server detected a syntax error when reading a CONFIG file.

In case the vendor Computer Associates product FLEE is active, a bug in FLEE can cause this error (they mix up the contents of the member).

To solve this, FLEE must be started after the z/VSE Connector Server Partition. Deactivating FLEE in the Server Partition does not help.

Codepage Problems with z/VSE Connectors and Navigator

For a correct display of special characters, like ä, ö, ü etc., the client codepage must match the codepage used by the z/VSE Connector Server.

For example, a client codepage of Cp1252 corresponds to the server ASCII codepage IBM-1252. The server side codepages are defined in the z/VSE Connector Server main CONFIG member like:

```
ASCII_CP     = IBM-850
EBCDIC_CP    = IBM-1047
```

The z/VSE Navigator shows the currently used client codepage when clicking on **Help->Show Codepage**. The client codepage is dependent on your country and keyboard settings of your workstation.

Problems with Logon to the z/VSE Connector Server

The z/VSE Connector Server does a RACROUT VERIFY call to check the user ID and password and to get the user's ACEE.

This implies that either the Basic Security Manager (BSM) or an External Security manager (ESM) like TopSecret must be active. The logon only works with z/VSE user IDs that are known be the Security Manager, i.e. users that are defined in the z/VSE control file.

There is a configuration member IESUSERS.Z (skeleton SKVCSUSR in ICCF Lib 59) which allows to restrict the access to the z/VSE Connector Server by IP address and/or user ID.

Problems with CA TopSecret

To be able to logon to the z/VSE Connector Server when the vendor Computer Associates product TopSecret is used, the z/VSE user IDs must be defined in TopSecret as shown in the output below of the TopSecret TSS transaction:

```
TSS LIST(SIE1) DATA(ALL)

ACCESSORID = SIE1      NAME       = JOHN DOE
TYPE       = CENTRAL   SIZE       =      512  BYTES
FACILITY   = *ALL*
CREATED    = 07/25/01  LAST MOD   = 10/04/01  09:29
PROFILES   = PROFGEN   PRFIRMSY
LAST USED  = 10/05/01 11:23 CPU(VSEA) FAC(BATCH   ) COUNT(00991)
-----------  SEGMENT CICS
OPCLASS    = 01
OPIDENT    = SYA
-----------  SEGMENT IESIS
IESFL1     = BAT,COD,VSAM
IESFL2     = BQA,ESC,COU,CMD,OLPD,XRM
IESINIT    = IESEADM
IESTYPE    = USERTYPE1,NEW,SELECT
-----------  ADMINISTRATION AUTHORITIES
RESOURCE   = *ALL*
   ACCESS  = ALL
ACID       = *ALL*
FACILITIES = *ALL*
LIST DATA  = *ALL*,PROFILES,PASSWORD,PWVIEW
MISC1      = *ALL*
MISC2      = *ALL*
MISC3      = SDT
MISC8      = LISTSTC,LISTRDT,REMASUSP,MCS,LISTSDT
MISC9      = *ALL*

TSS0300I  LIST     FUNCTION SUCCESSFUL
```

Here is a sample command to add the necessary attributes to a user:

```
TSS ADD(user-acid) IESINIT(IESEADM) IESTYPE(USERTYPE1,NEW,SELECT)
   IESFL1(BAT,PSL,COD,VSAM) IESFL2(BQA,ESC,COU,CMD,OLPD,XRM)
```

Problems Accessing VSAM Catalogs

There can be two error situations when accessing VSAM catalogs from the z/VSE Navigator or other z/VSE Connector clients:

- There are no catalogs shown at all.

  This means that the VSAM.MASTER.CATALOG (IJSYSCT) can not be opened. This may be caused by a invalid label information for IJSYSCT. If the customer has a IJSYSUC in the standard labels which is not pointing to the master catalog, the customer should apply PTF UQ54777 which fixes this problem, or add the following DLBL to the startup job:

  ```
  // DLBL IJSYSCT,'VSAM.MASTER.CATALOG',,VSAM,CAT=IJSYSCT
  ```

- All catalogs are shown, but accessing a specific user catalog fails.

  This may be caused by a invalid label information for this catalog.

In both cases running an LSERV is helpful.

Problems of Sharing an Active VSAM File

The latest changes made to a VSAM file may not always be found with a read or get request. A 'no record found' or older copy of the record can be indicative of this problem. This is especially critical in applications that use multiple strings or tasks, as with CICS. Share option 2 does not provide read integrity - such a read does not do I/O to disk if a copy of the target CI is currently in a buffer. if the CI in the buffer is 'downlevel' or an older copy of the CI that is actually on disk, then the latest record or correct record may not be returned to the application. Direct reads (or random reads) with share option 4 files will do I/O to disk even if there is a copy of the target CI in a buffer. This insures that whatever is on disk will be returned.

This however does not totally insure read integrity if the latest record is still sitting in a buffer somewhere. A read that is a 'get-for-update' will force buffers to be written to disk that need to be (with I/O pending), but this of course only affects buffers that the task has control over: it will not affect buffers that are left in storage from an update to a base cluster when the read is through a path nor will it affect buffers in other partitions. Read integrity is also not guaranteed with sequential processing even if the file is share option 4, unless a get-for-update is done. Under CICS this is accomplished when the file is open for output or the update request is specified even though update requests are never made (only gets or browses). If buffers still contain records that need to be written to disk and another task needs to see these records but has no access to these buffers, as the case is when using a path via an aix to read records that have been updated with the base, nothing can be done with the read task to see the latest updates or changes. the task which updated the file must force the buffers to be written to disk before the other task can see them. Buffers can be written to disk with assembler tclose or endreq macros, when the file is closed, or when the task does another request which requires a new buffer to be used (this forces the contents of the old buffer to be written to disk if it had an I/O pending).

Summary.
If a user reports read integrity problems that are resolved after the file is either closed or after other records are accessed in between the unsuccessful and successful read, then share option 4 may help them. If paths and aix's are used to read records that have been updated via the base then they must be aware that these records must be forced to disk via the update task before the read task will see them, This can be accomplished by causing that task to go to end-of-task.

Error Message IESC1005E CANNOT SET UP TCP/IP LISTENER

The error message can have several causes:

- TCP/IP for z/VSE has not been started.
- TCP/IP for z/VSE has been started with a system ID, where the system ID is specified like:

  ```
  // EXEC IPNET,SIZE=IPNET,PARM='ID=nn,INIT=.....
  ```

  The default is 00.

  To use the TCP/IP services from within another partition this partition has to 'know' the system ID. This is specified as follows:

  ```
  // OPTION SYSPARM='nn'
  ```

  where nn is the system ID.

  Customers may use the skeleton SKVCSSTJ in ICCF lib 59 to add this statement.

- The $EDCTCPV phase is missing or the wrong phase gets accessed.

  The $EDCTCPV phase is part of TCP/IP for z/VSE and implements the C-socket interface, which is used by the z/VSE Connector Server. This phase is also part of the LE-runtime library as a dummy phase. So check your LIBDEFs and make sure the right phase (the bigger one) gets accessed. The LE-runtime library is normally in PRD2.SCEEBASE, TCP/IP is normally in PRD2.TCPIPC.

Abend AEZC when using VSE SOAP Support

When you receive an Abend AEZC when using VSE SOAP Support, this indicates that the user program (that is implementing the real service) which is called by the VSE SOAP enginge or by the proxy code, is AMODE24. The CICS Web Support transaction CWBA runs with TASKDATALOC(ANY).

You need to either change your program to AMODE 31 or change transaction CWBA to TASKDATALOC(BELOW) (better use another alias transaction name).

Problems with VSE SOAP Support after installing APAR PK18932/PTF UK11718 or superseeding.

The Connector APAR PK18932/PTF UK11718 has a soft prereq on 2 LE APARs:

- PK20013
- PK19358

PK20013 correct the LE/C supplied CSD definitions to include iconv() function modules EDCUCONV and EDCUCNVI. Besides applying the PTF, please ensure that programs "EDCUCNVI" and "EDCUCONV" are defned to GROUP(CEE) in CICS System Definition (CSD) file, or that program autoinstall is activated.

If these 2 programs are not known by CICS, or can not be autoinstalled, the codepage convertion done in VSE SOAP engin will not work. An empty SOAP response can be the result.

ACTION: User's which have the CICS AUTOINSTALL FEATURE disabled must rebuild/refresh CICS CSD GR(CEE) e.g. via submitting CEECCSD (ICCF62) or SKLE370 (ICCF59). A subsequent CICS cold start or CEDA INSTALL GR(CEE) should be performed to activate the CSD changes.

J2CA0294W: Deprecated usage of direct JNDI lookup of resource eis/VSEConnector

Customer is getting the following Message in WebSphere App Server when using VSE Connetcor Client with JNDI Lookup:

```
[4/25/07 8:47:57:262 MDT] 00000045 ConnectionFac W   J2CA0294W:
Deprecated usage of direct JNDI lookup of resource eis/VSEConnector.  The
following default values are used: [Resource-ref settings]
```

Please see here for more information about that message:
http://www.ibm.com/support/docview.wss?uid=swg21220886

As a summary, you need to use a JNDI name like "java:comp/env/eis/VSEConnector" in your servlets (instead of just "eis/VSEConnector"). This makes it a indirect JNDI name.

VSE Connector Server causes high CPU utilization when using BSI TCP/IP (Barnard)

Cusomers reported a very hight CPU usage (50% Server, 50% TCP/IP Stack) when downloading a VSAM file using VSE Connector Server when using Barnard TCP/IP Stack (BSI).

The reason seems to be an different behaviour regarding select() and send() socket calls. Adding the following option into the startup job for VCS seems to help:

```
// SETPARM SENDALL='YES'
```

Message: The server returned a invalid or unsupported version number.

If you get a message (Java Exception) "The server returned a invalid or unsupported version number" this indicates that there is an mismatch between the version of the server code on VSE and the client code on the PC/Workstation. Usually the customer has upgraded the VSE version/release, but did not upgrade the client code on thge PC/Workstation. In general, newer clients can work with older servers, but not the other way round.

It is suggested to download and install the newest client level from http://www.ibm.com/systems/z/os/zvse/downloads/

Running VSAM Redirector Server or VTAPE Server in background under Linux

Use the following command to start the server:

```
./run.sh 1>stdout.txt 2>stderr.txt </dev/null&
```

Running VSAM Redirector or VTAPE as a Windows Service

BM provides several server applications for use with z/VSE that are implemented in Java:

- VSAM Redirector Server
- VSE VTAPE Server
- VSE Script Server

Users may wish to run such a server on an unattended Windows system. They may want to keep these server applications running even if no user is signed-on on Windows. The JavaService tool allows running a Java application as a Windows service in the background. It acts as a wrapper inbetween the Windows Service Control Manager and the Java Program.

Download and extract the ZIP file into a new directory (e.g. C:\Temp). A Readme.pdf file is contained in the ZIP file with further installation instructions.

The JavaService Tool is provided 'As is', no official support, no warranty: http://www.ibm.com/systems/z/os/zvse/downloads/tools.html#jav

Error: 'ConverterFactory ERROR: The default codepage <'Cp1047'> is not supported'

See http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=6329080

The charsets.jar file is an optional feature of the JRE. To install it, you must choose the "custom installation" and select the "Support for additional locales" feature. It is selected by default on some machines.

List of supported charsets: http://java.sun.com/javase/6/docs/technotes/guides/intl/encoding.doc.html

Error: 'java.io.UnsupportedEncodingException: Cp1047 - charsets.jar missing'

See http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=6329080

The charsets.jar file is an optional feature of the JRE. To install it, you must choose the "custom installation" and select the "Support for additional locales" feature. It is selected by default on some machines.

List of supported charsets:
http://java.sun.com/javase/6/docs/technotes/guides/intl/encoding.doc.html

VSAM Redirector version z/VSE V4.1: "SYSIBM:CLI:-805". SQLSTATE=38553

When using the VSAM Rediretcor DB2Handler of Version z/VSE 4.1 or later, you may get the following error:

```
FFSTORES,1 ------------------------------------------------------------
FFSTORES,1 Database column information for table 'nnnn':
FFSTORES,1 Error (getMapInfo): COM.ibm.db2.jdbc.DB2Exception: IBM.CLI
Driver.DB2/NT SQL0443N  Routine "SYSIBM.SQLCOLUMNS" (specific name
"COLUMNS") has returned an error SQLSTATE with diagnostic text
"SYSIBM:CLI:-805".  SQLSTATE=38553
```

Solution: Rebind the @db2schema.bnd file in DB2 as described here:
http://www.ibm.com/developerworks/db2/library/techarticle/dm-0509zikopoulos/

Performance problems due to Delayed Ack

Because of the Delayed Ack problem, many applications that use the LE/C socket interface may run into performance problems:

- slow transfers in general due to 200msec waits per send call
- Throughput breakdown when multiple connections are established to same server

There is a fix for that probem in TCP/IP for z/VSE, but you need to activate it. Per default it is deactivated.
You activate it by runnig the following job:

```
* $$ JOB JNM=$SOCKOPT,CLASS=A,DISP=D
* $$ LST CLASS=A
// JOB $SOCKOPT
// OPTION CATAL
// LIBDEF *,SEARCH=tcpip.lib
// LIBDEF *,CATALOG=tcpip.lib
*
// EXEC ASMA90,SIZE=ASMA90,PARM='SZ(MAX-200K,ABOVE)'
         PUNCH    ' PHASE $SOCKOPT,* '
$SOCKOPT CSECT
*
* * This phase is used by the BSD-C interface to allow global options
* * that affect the operation of all sockets in a partition.
         SOCKOPT  CSECT,                   Generate a csect            X
               BSDCFG1=$OPTMECB+$OPTSNWT,  Socket options flag         X
               CLST=-1,                    Seconds to wait for close   X
               CSRT=59,                    Seconds before socket reuse X
               SSLLIBN=KEYLIB,             SSL library name            X
               SSLSUBN=SSLKEYS,            SSL sublib name             X
               SSLMEMN=MYKEY512,           SSL member name for keys    X
               SSLDEBG=00,                 SSL debugging flag          X
               SSLFLG1=00, 80=req_close_notify_alert                   X
               SSLFLG2=80, 80=do not use HW-Crypto                     X
               SSLSTOR=80,                 SSL storage useage          X
               SYSID=00                    Use this TCP/IP sysid
*
         END   $SOCKOPT
/*
// EXEC LNKEDT,PARM='MSHP'
/*
/&
* $$ EOJ
```

How to activate an API and IP trace with Barnard TCP/IP Stack (BSI)

API trace: Add // SETPARM IPTRACE='YYNY' to your applications JCL. Trace output will go to your applications SYSLST.

Packet trace: a.b.c.d = IP address of remote host

MSG BSTTINET,D=TRACEIP a.b.c.d 255.255.255.255 MSG BSTTINET,D=IP TRACE2 ON test MSG BSTTINET,D=IP TRACE2 OFF MSG BSTTINET,D=SEGMENT * $$ LST CLASS=...

Tracing the z/VSE Connector Server

To get a full trace of the z/VSE Connector Server, issue the following command at the console:

```
MSG nn,DATA=SETTRACE DD:SYSLST 0xFFFFFFFF
```

In this case the trace goes to SYSLST. You may also specify SYSLOG to have the trace messages on the console.

z/VSE Connector Books

The following book is available and can be downloaded from the z/VSE e-business Connectors and Utilities Internet page in PDF format:

z/VSE e-business Connectors User's Guide.

More Information and FAQs

For more information, FAQs, and downloading latest code see our z/VSE e-business Connectors and Utilities home page at

`http://www.ibm.com/systems/z/os/zvse/products/connectors.html`

Chapter 21. VSE Health Checker

The VSE Health Checker is a Java tool that retrieves and evaluates many VSE system parameters. The tool is based on the VSE e-business connectors, i.e. needs the VSE Connector Client installed on the workstation and the VSE Connector Server running on VSE.

You can download all necessary components from the VSE homepage.

`http://www.ibm.com/systems/z/os/zvse/downloads/`

When running the Health Checker, it takes a snapshot of all considered system parameters. This process may take up to several minutes. However, additional CPU load is very low.

After the data retrieval has finished, you can evaluate the data via the graphical user interface. An HTML-based report can be stored for further use.

Also gathered data can be stored in XML format. This allows to load the data again at any time without running the data retrieval process again. It also allows to send gathered data to IBM for further investigation.

Documentation

Documentation is provided together with the tool as HTML-based help. There is a "quick start" help section to show you how to do the first steps with the tool.

There is an article about health checking in VSE in the zJournal from June 2006. Read it on: *http://enterprisesystemsmedia.com/article/health-checking-for-z-vse* .

Dependencies

The VSE Health Checker has the following prerequisites.

- TCP/IP running on VSE
- VSE Connector Server running on VSE (job STARTVCS)
- VSE Connector Client installed on workstation side
- Java 1.5 or higher
- VSE System class library vsesystem.jar (part of the Health Checker package), which is also available as separate package from the VSE Homepage
- The STAT transaction must be defined in order to obtain CICS TS status data
- The CHKT transaction must be defined in order to obtain a list of TS queues

Restrictions and Known Problems

Currently the following restrictions apply.

- The TCP/IP plugin only supports parameters of the CSI TCP/IP stack. Other VSE TCP/IP stacks can be used to connect to VSE and retrieve data, but parameters cannot be displayed.

- The Health Checker relies on parsing the output of console commands and VSE/POWER jobs. When using OEM products like MLOG, which modify the output of commands, the Health Checker is no longer able to parse the output. See help section MLOG considerations for how to solve the problem. But there is a function to specify a prefix and suffix to console commands with any user defined string. This can be used to force the original format of console output. See General Options for details.
- Security Manager related parameters are only supported for the IBM provided Basic Security Manager (BSM). External Security Managers are currently not supported.
- When list queue output is automatically spooled to a VM/CMS user, the Health Checker can't access this output for parsing. Example CICS statistics.

Some Usage Hints

Timeout values

The Health Checker gets its data by submitting VSE/POWER jobs, reading VSAM files, and entering operator console commands.

With console commands there is sometimes a problem to determine the end of a particular console output. For example, when issuing a MAP BG command, the console output always ends with

```
AR 0015 1I40I  READY
```

When issuing a D DYNC, there is no such indication. Example:

```
AR 0015 1C39I COMMAND PASSED TO VSE/POWER
F1 0001 1Q6AI  ******   ACTIVE DYNAMIC CLASS TABLE DTR$DYNC.Z    ******
F1 0001 1Q6AI  CLS STATE   ACT/MAX ALLOC    SIZE  SP-GETV  PROFILE LUBS
F1 0001 1Q6AI   C  ENAB      0  9     1M    500K    128K   STDPROF   50
F1 0001 1Q6AI   P  ENAB      0 32     1M    512K    128K   PWSPROF   50
F1 0001 1Q6AI   R  ENAB      1  3     8M   1024K    128K   STDPROF  100
F1 0001 1Q6AI   S  ENAB      0  2    15M   1024K    128K   STDPROF  100
F1 0001 1Q6AI   Y  ENAB      0  8     3M   1024K    128K   STDPROF   50
F1 0001 1Q6AI   Z  ENAB      1  3     5M   1024K    128K   STDPROF   50
```

Therefore, in the latter case, the Health Checker waits until a timeout period is reached.

As such timeout values are system dependent, the values can be set via the General Options dialog in the Health Checker GUI. There you can tune the values for your system.

Chapter 22. Debug Tool for VSE/ESA (LE)

This chapter gives some hints on how to use the Debug Tool for VSE/ESA. It does NOT contain detailed information. For further questions about installation and usage, please refer to the documentation listed below. For information on enhancements available with mod-level 1.1.1, see the memo supplied with the product.

Documentation Reference

Following documentation is available for the IBM Debug Tool for VSE/ESA :

Debug Tool for VSE/ESA 1.1.0

SC26-8798-00 Debug Tool Installation and Customization Guide

An accumulation of all changes to the book until the next edition is released is kept in RETAIN InfoAPAR II10023.

SC26-8797-00 Debug Tool User's Guide and Reference

An accumulation of all changes to the book, until the next edition is released, is kept in the RETAIN InfoAPAR II10286.

Debug Tool for VSE/ESA 1.1.1

Refer to the above documentation the IBM Debug Tool for VSE/ESA 1.1.0 as well as informational APARs II10023 and II10286 available from your local IBM support center.

Debug Tool Run-Time Environment

Debug Tool for VSE/ESA 1.1.0

Debug Tool requires the LE/VSE 1.4.0 run-time library (or later), and in particular the LE/VSE C-specific run-time library. This is achieved installing the VSE C Language Run-time Support feature of VSE/ESA Version 2 Release 2 or above.

If you want to write and debug application programs in COBOL/VSE or PL/I VSE, the LE/VSE run-time library components matching the language used must be installed, too.

Debug Tool for VSE/ESA 1.1.1

Debug tool for VSE/ESA 1.1.1 requires the LE/VSE Version 1 Release 4 Modification Level 1 (or later) run-time library and in particular the LE/VSE C specific run-time library. This is installed by default with VSE/ESA 2.5 and above. If you need to debug application programs written in COBOL or PL/1 you will also need the COBOL or PL/1 specific components of the LE/VSE 1.4.1 (or later) run-time library.

VTAM Considerations

Planning to use the tool to debug your applications interactively, you must add the Debug Tool VTAM definitions to your system by following the instructions on Page 18 of 'Debug Tool for VSE/ESA Installation and Customization Guide'.

IBM supplied job EQAWAPPL holding the predefined VTAM application names, should be used for enabling an interactive debug session. Otherwise the MFI-suboption of LE/VSE run time option TEST cannot be activated.

This member is available in ICCF library 62 after installation of Debug Tool for VSE/ESA.

CICS Considerations

If you plan to use the Debug Tool with CICS applications, you must update your CICS system following the instructions on Page 19 of 'Debug Tool for VSE/ESA Installation and Customization Guide'.
All necessary steps are described in that manual.
Below the needed updates in the CSD file are explicitely listed. You can follow these steps, if you missed them in the installation process or when you want to configure a second CICS/TS.

To establish the Debug Tool related program definitions in the CICS CSD file assistance is available by using the IBM provided job skeletons and Z-Books.

The CSD file must hold the information about :

- the programs comprising the Debug Tool

  EQATCSD.Z should be used to do the update; it is also available in ICCF library 62.

- the C Run Time Library

  Job skeleton SKLE370 (in ICCF library 59) referring to EDCCCSD.Z (in PRD2.SCEEBASE) should be used to do the update

- the LE/VSE Common Run Time Library

  SKLE370 referring to CEECCSD.Z should be used to do the update

- the Language Specific Run Time Library for COBOL and/or PL/I, in case your applications are coded in those programming languages.

  SKLE370 referring to IGZCCSD.Z (for COBOL) and IBMCCSD.Z (for PL/I) should be used to do the update.

Note: In SKLE370 the utility DFHCSDUP is used. Be aware that

```
// EXEC DFHCSDUP,SIZE=300K
```

will lead to an error. You should use

```
// EXEC DFHCSDUP,SIZE=DFHCSDUP
```

See “DFHCSDUP Utility” on page 307

For details about the Run Time Libraries see *LE/VSE Installation Customization Guide.*

Member EQATCSD also provides the necessary support to run the Debug Tool transaction DTCN . This member can be either found in ICCF library 62 or in 'Debug Tool for VSE/ESA' installation sublibrary PRD2.PROD.

In general there are three mechanisms available to invoke the Debug Tool under CICS :

- Debug Tool CICS Interactive Utility via transaction DTCN. This transactions allows you to specify all debugging related requirements. You can specify terminal-ids for the debug session, supply the transaction-id of the application you wish to debug, and you can specify run-time options (e.g. TEST option) to override the active LE/VSE CICS run time options dynamically.

  On the application side an INCLUDE EQADCCXT statement is required for inclusion during the link step of building an executable phase. Adding this user exit will be sufficient to invoke DTCN.

  This is the recommended way to activate the Debug Tool under CICS. For details see *Debug Tool User's Guide and Reference*, page 107 ff.
- Linking a CEEUOPT module into the application, containing the appropriate TEST option. This tells LE/VSE to invoke the Debug Tool for VSE/ESA every time the application is run.
- Run-time directive #pragma runopts(test) for C, or the PLIXOPT string for PL/I or CALL CEETEST within the application.

VSE Partition Requirements

The minimum recommended partition size for batch programs running with the Debug Tool is 8 MB. However, 6MB may be sufficient if the LE/C SVA eligible modules are resident in the SVA. Batch programs can run with the Debug Tool in either a static or dynamic partition. The partition in which an interactive the Debug Tool session starts is unavailable to other users for the duration of the Debug Tool session.

General Remarks for Activating Debug Tool for VSE/ESA

This is the general principle to invoke the Debug Tool for VSE/ESA.

1. Either - compile the LE/VSE Conforming Application with compile option TEST,

   or - code library services like CEETEST, PLITEST or ctest() function within your application program to generate calls to the Debug Tool.
2. Run the LE/VSE Conforming Application with TEST run-time option to start a debugging session.

Here are some general attributes of the Debug Tool for VSE/ESA you should be aware of.

- The VTAMLU (running in batch mode), or the terminal-id (running under CICS), where the debugger window is supposed to show up, must not be in session with another application when the debug session is started. For TCPIP telnet terminals see the information on “Debug Tool for VSE/ESA 1.1.1” on page 337

- Debug Tool for VSE/ESA is operating on compiler listings and must have access to them. Exception: using C for VSE/ESA, the Debug Tool needs access to the program source.

  Therefore those files must be stored and available when activating the Debug Tool.
- Debug Tool will check on compile units via accessing SAM, VSAM/SAM files or AF-members in the Librarian 'library sublibrary' hierarchy.
- There is a predefined printer exits EQALIST which assists the user in storing compiler listings during compile time.
- SYSLST assignment to a SAM file prior to running the compile unit is an alternative, too. In this case the SAM file must be part of the JCL specified in the form USERID.CUNAME.LIST, where CUNAME is the name of the compile unit.

Debug Tool Customization and Verification Jobs

This is the list of members punched to ICCF library 62 during installation of the Debug Tool for VSE/ESA via the Interactive Interface. These jobs will assist you in various verification and customization steps.

EQATCSD	Debug Tool - CICS CSD Definitions for CICS/TS
EQACCSD	Debug Tool - CICS CSD Definitions for CICS/VSE
EQACPCT	Debug Tool - CICS PCT Definitions
EQACPPT	Debug Tool - CICS PPT Definitions
EQAWAPPL	Debug Tool - Sample VTAM APPL Book
EQAWIVC1	Debug Tool Example - Verify COBOL/VSE batch program
EQAWIVC2	Debug Tool Example - Verify COBOL/VSE batch interface
EQAWIVC3	Debug Tool - Assemble runtime options for EQAWIVC4
EQAWIVC4	Debug Tool Example - Verify COBOL/VSE CICS interface
EQAWIVH1	Debug Tool Example - Verify C/VSE batch
EQAWIVH2	Debug Tool Example - Verify C/VSE batch interface
EQAWIVH3	Debug Tool Example - Verify C/VSE with CICS
EQAWIVP1	Debug Tool Example - PL/I VSE batch
EQAWIVP2	Debug Tool Example - PL/I VSE batch interactive
EQAWIVP3	Debug Tool Example - PL/I VSE CICS
EQAWUGC1	Debug Tool Example - COBOL Financial Sample Program
EQAWUGH1	Debug Tool Example - C/VSE Simple Calculator
EQAWUGP1	Debug Tool Example - PL/I VSE Simple Calculator

Figure 136. Debug Tool Customization and Verification Jobs

Limitations of Debug Tool for VSE/ESA

Debug Tool for VSE fully supports the process of debugging multi-language applications. There are no limitations when all the programs in the application were compiled with an LE-conforming compiler.

However, if the application contains programs compiled with non- LE-conforming compiler, the Debug Tool will not support source-level debugging of those program units.

The application will execute unhindered while control is within a program written in an unsupported language. the Debug Tool will regain control and allow debug commands to be issued once execution returns to a program unit written in a supported language, and compiled with TEST option.

Debug Tool for VSE does NOT support the cooperative mode of operation where the program being debugged is executing on a VSE/ESA host system and the Debug Tool is executing on a programmable workstation (PWS).

Debug Tool for VSE cannot execute without an active LE/VSE environment, Release 1.4 or later.

Internet Link to Debug Tool for VSE/ESA 1.1.0 / 1.1.1

For a small introduction to the interactive debugger you might like to visit WEB-page

http://www.ibm.com/software/awdtools/debugtoolvse/

Debug Tool Related Service

Debug Tool for VSE/ESA 1.1.0

Before installing the Debug Tool for VSE/ESA, it is recommended to check via IBM Level 1 Support for fixes contained in the VSE Service Buckets DTVSE110, VSELE140, and CVSE110. For users running the Debug Tool for VSE/ESA 1.1.0 with VSE/ESA 2.5 and LE/VSE 1.4.1, APAR PQ39609/PTF UQ48821 is required

Debug Tool for VSE/ESA 1.1.1

Before attempting to use the Debug Tool for VSE/ESA 1.1.1, it is recommended that you contact IBM level 1 support for fixes contained in the VSE service buckets DTVSE111, VSELE141 and VSEESA25x or VSEESA26x. Here is a list of recommended fixes for the Debug Tool known by Oct,2002:

- PQ66260
- PQ67032
- PQ67266
- PQ68277
- PQ74981

Debug Tool Related Usage Problems

Debug Tool for VSE/ESA 1.1.0

This is an overview about most common usage problems with the Debug Tool for VSE/ESA :

1. Not having installed the LE/VSE specific C component 5686-094-08 (and optionally 5686-094-09 for Japanese support). This will lead to various problems as the Debug Tool depends on this run-time support.
2. Even if having installed the C component the associated CICS CSD-file entries (or alternatively CICS PPT program definitions) must be in place. A CICS coldstart should always be done before relying on this support. Use skeleton "SKLE370" in ICCF lib 59 for such enablement.
3. Debug Tool and Service. Usually PTFs for the Debug Tool will replace OBJ-type members, and therefore require access to LE/VSE sublibrary PRD2.SCEEBASE when re-linking new control modules, i.e. phases.

   Therefore please make sure sublibrary PRD2.SCEEBASE is included in the PERMANENT SEARCH LIBDEF chain running service upgrades. Otherwise the link output will show unresolved external references.
4. "Batch Interactive" Debug Tool applications e.g. installation verification program EQAWIVC2 (ICCF lib 62) rely on having exclusive access to the terminal defined via the TEST runtime option (and MFI suboption for VTAM LU id). Therefore the terminal where you expect the Debug Tool window to pop up must NOT be busy with other applications. Dial to such terminals via explicitly specifying the "cuu" address and keep the following VTAM application screen you get directed to untouched - do NOT enter CICS!

   Entering CICS on the terminal may show you the LU-id needed for the MFI suboption of TEST run-time option, PF4 will bring you back to the VTAM screen.
5. When using TELNET terminals for BATCH interactive debug tool sessions, the currently available CSI TCP/IP stack cannot support multiple sessions on one TELNET DAEMON. When a TELNET DAEMON is in session with another application (eg CICS), TCP/IP is not aware that other applications (like the Debug Tool) may request a secondary session with the same TELNET DAEMON. The secondary session may be established, but there are problems using it. These are :
   a. SNA logmodes are not supported by the TELNET DAEMON
   b. When the other application terminates its secondary session with the TELNET DAEMON , the primary session is dropped as well. When using Telnet terminals with the Debug Tool it is recommended that a non-SNA logmode is first in the active logmode table. The VTAM definition for a telnet terminal used with the Debug Tool must have at least 2 sessions defined. This is because telnet terminals may be in session with another application (eg CICS) since telnet terminals cannot be at a USSTAB screen which is normally required by the Debug Tool for interactive debugging mode.
6. "CICS Interactive" applications like EQAWIVC3 / 4 also rely on having exclusive terminal access. Nevertheless they require a ready CICS session e.g. a vailable via PF6 selection on the main entry panel of the "Interactive interface".

Note the difference to the batch interactive debug scenario, needing the VTAM screen.

7. Never forget to catalog VTAM-book EQAWAPPL.B in PRD2.CONFIG sublibrary (which is already part of the VTAM startup LIBDEF chain). Otherwise the Debug Tool interactive sessions can't be enabled via console command "V NET,ACT,ID=EQAWAPPL".

   In case the current status is unsure use "D NET,APPLS" command in order to see whether the Debug Tool definitions are part of the customers system.

8. If the Debug session can't be established, e.g. the VTAM ACB could not be opened due to missing activation of EQAWAPPL, the input usually read by the Debug Tool remains on SYSIPT, and VSE Job Control will finally end up with message

   1S01D INVALID STATEMENT

   when trying to interpret it as JCL statements.

Debug Tool for VSE/ESA 1.1.1

This is an ovreview of the most common usage problems associated with using the Debug Tool for VSE/ESA :

1. Not having the LE/C run-time component installed. Ensure that component 5686-066-33-R55L for VSE/ESA 2.5.x or 5686-066-33-R65L for VSE/ESA 2.6.x is installed.
2. For problems under CICS, ensure that message CEE3550I is issued at CICS startup to confirm that the required LE/C run-time specific components are installed and active in the CICS system.
3. Applying the Debug Tool PTFs require access to the LE/VSE installation library. Therefore it is recommended that the LE/VSE installation sublibrary is placed in the partition permanent search chain so that it will be available for MSHP PTF applies.
4. When using the Debug Tool in BATCH interactive mode, ensure the terminal defined in the MFI statement is not in session with another application. The terminal here you expect the Debug Tool session to pop up should be displaying a VTAM USSTAB display.
5. When using TELNET terminals for BATCH interactive debug tool sessions, the currently available CSI TCP/IP stack cannot support multiple sessions on one TELNET DAEMON. When a TELNET DAEMON is in session with another application (eg CICS), TCP/IP is not aware that other applications (like the Debug Tool) may request a secondary session with the same TELNET DAEMON. The secondary session may be established, but there are problems using it. These are :

   a. SNA logmodes are not supported by the TELNET DAEMON
   b. When the other application terminates its secondary session with the TELNET DAEMON , the primary session is dropped as well. When using Telnet terminals with the Debug Tool it is recommended that a non-SNA logmode is first in the active logmode table. The VTAM definition for a telnet terminal used with the Debug Tool must have at least 2 sessions defined. This is because telnet terminals may be in session with another application (eg CICS) since telnet terminals cannot be at a USSTAB

screen which is normally required by the Debug Tool for interactive debugging mode.

6. "CICS Interactive"applications like EQAWIVC3 /4 also rely on having exclusive terminal access. Nevertheless they require a ready CICS session e.g.available via PF6 selection on the main entry panel of the "Interactive interface".Note the difference to the batch interactive debug scenario,needing the VTAM USSTAB screen.
7. Never forget to catalog VTAM-book EQAWAPPL.B in PRD2.CONFIG sublibrary (which is already part of the VTAM startup LIBDEF chain).Otherwise the Debug Tool interactive sessions can't be enabled via console command "VNET,ACT,ID=EQAWAPPL". In case the current status is unsure use "D NET,APPLS"command in order to see whether the Debug Tool definitions are part of the system.If the Debug session can't be established,e.g.the VTAM ACB could not be opened due to missing activation of EQAWAPPL,the input usually read by the Debug Tool remains on SYSIPT,and VSE Job Control will finally end up with message
when trying to interpret it as JCL statements.

Chapter 23. Miscellaneous Hints and Tips

These are some hints and tips that did not fit into one of the previous sections.

How to get Control in BG during System Startup

Sometimes it is necessary to get control in BG before the very first job card is executed. Typically this is true if the hard copy or recorder file needs to be re-created. This can only be done before job control has opened such files. Opening of these files, however, happens with the first job card.

This describes a method to get control in BG before job control starts allowing to enter any create command for e.g. recorder or hard copy file. Follow theses steps:

1. interrupt a normal IPL by specifying the IPL load parameter LOADPARM ..P
2. enter your usual IPL procedure name, but for the JCL name enter a name starting with '$$' of a procedure that does not exist
3. IPL will now start until message

   ```
   BG-0000 1N20D PROCEDURE NOT FOUND
   ```

   comes up.
4. now enter any command that needs to be entered before job control has started

How to Re-Format the Recorder or Hardcopy File

The recorder or hardcopy file can only be recreated (formatted) during IPL before the first job card was executed. After the first job card was executed these files are opened and cannot be formatted any more.

See “How to get Control in BG during System Startup” how to get control in BG during IPL before the first job card is executed. Then enter the following to recreate the recorder file:

1. after system message

   ```
   BG-0000 1N20D  PROCEDURE NOT FOUND
   ```
2. enter **'0 sysdef dspace,dsize=15m'**
3. enter **'0 vdisk unit=fdf,blks=2880,volid=vdidla,usage=dla'**
4. enter **'0 exec proc=stdlabel'**
5. system responds

   ```
   BG 0000 EOP STDLABEL
   BG 0000 1I00D  READY FOR COMMUNICATIONS
   ```
6. enter **'0 set rf=create'**
7. system responds

   ```
   BG-0000 0D16D READY FOR COMMUNICATIONS
   ```
8. enter '0 // JOB XXXX'
9. system responds

```
BG-0000 4433D EQUAL FILE ID IN VTOC IJSYSRC
              SYSREC = 159 SYSWK1 VSE RECORDER FILE
```

10. enter '0 delete', now formatting begins and control is returned to user after file was formatted
11. now IPL normally

The same procedure can be used to recreate the hardcopy file. Use

'0 set hc=create'

instead.

Important:

Since labels are required for the formatting of the recorder file or hardcopy file, the label area on VDISK has to be created first. The specifications above are those from the standard BG JCL ASI procedure $0JCL. If you have modified the SYSDEF DSPACE or VDISK specifications then use your own values.

If the hardcopy or recorder file must be moved to another volume, be aware that both files must reside on the same volume. If you need to move one file then both files must removed. The IPL DEF SYSREC=cuu must also be changed along with the STDLABEL info for both files.

SDAID GETVIS/FREEVIS TRACE

Is opposed to a simple SVC trace (3E,3D) which only records the execution of instructions, this new trace type offers now the capability to record the full results of all GETVIS/FREEVIS requests as provided by the VSE GETVIS/FREEVIS supervisor routines.

GETVIS TRACE command

```
TRACE GETVIS=target SUBPOOL=name LOCATION=region
```

target = PARtition

SPAce

SVA

name = SUBPOOL NAME. If ommited all subpools are traced. If the

subpool name contains characters which have to be treated

treated as hex-characters, enclose them in <..>.

E.G: SUBPOOL=MYPOOL

SUBP=INLC<0021>

region = BELow Trace only requests in 24 bit getvis area

ANY Trace requests in 24/31 bit getvis area

if omitted, ANY is assumed.

All other TRACE parameters like OUTPUT, LOCATION etc are still valid.

You may invoke SDAID in PROMPTING MODE to get a step by step guidance when entering "15 ?" to the systems response.

EREP Hints and Tips

EREP (Environmental Record Editing and Printing Program) is a Program Product which is available for z/OS, z/VM and z/VSE. EREP provides a batch utility program to print formatted reports from the records placed in the error recording data set (ERDS) by the error recovery program (ERP) of the operating system. On z/VSE this ERDS file is commonly named **recorder file**.
The VSE recorder file resides on a disk. Its location is defined

- by the IPL DEF Statement, for example:

```
DEF SYSREC=SYSWK1
```

- and by a disk label, for example

```
// DLBL IJSYSRC,'VSE.RECORDER.FILE',99/366,SD
// EXTENT SYSREC,SYSWK1,1,0,8415,60
```

The VSE recorder file may reside on a shared DASD, but the file itself cannot be shared. Each VSE system must have its own recorder file extent.
Records in recorder file are written by the VSE System Tasks RAS and ERP (ERP may do this on request from other VSE components). When VSE runs as a VM guest, VSE recording into its recorder file depends on the VM CP Definition **SET SVC76**.
If **SET SVC76 CP** has been defined VM CP will intercept the recording requests, eventually record them into its own ERDS, and post VSE not to record them into the VSE recorder file.
With **SET SVC76 VM** VM CP will not intercept these requests.

EREP Reports

EREP reports from the VSE recorder file are produced with a batch job:

```
// EXEC IFCEREP1
report parameters
selection parameters
processing parameters
/*
```

with (selected) report parameters:

```
PRINT=AL    requests all the detail reports
PRINT=NO    requests that no reports be generated at all.
PRINT=PS    requests detail edit and detail summary reports.
(selected) selection parameters:
TYPE=code   e.g: M for Machine check records,
                 C for Channel check/channel report word/subchannel
                   logout
DATE=yyddd[-yyddd]
CUA=cuu1[-cuu2]
(selected) processing parameters:
ACC=Y|N     Copy records to output data set ("history file") on
            tape/disk
ZERO=Y|N    Clear the recorder file
MERGE=Y|N   Use history file plus recorder file for processing
```

(For details on these and other parameters see: EREP User's Guide and EREP Reference Manual).

EREP Sample Jobs:

Print and clear the recorder file

```
// EXEC IFCEREP1
   ACC=N                                    <- No History
   PRINT=PS
   ZERO=Y                                   <- Clear recorder file
/*
```

Figure 137. EREP: Print and clear the recorder file

Merge Recorder File content into a "history file"

and clear the Recorder File

```
// DLBL HISTIND,'VSE.EREP.HISTOD'           <- Input file
// EXTENT SYS008,SYSWK1,1,0,8620,30
// ASSGN SYS008,DISK,VOL=SYSWK1,SHR
// DLBL HISTOD,'VSE.EREP.HISTOD'            <- Output file
// EXTENT SYS009,SYSWK1,1,0,8620,30
// ASSGN SYS009,DISK,VOL=SYSWK1,SHR
// EXEC IFCEREP1
   ACC=Y                                    <- Create new History
   MERGE=Y                                  <- Merge old and new
   PRINT=PS
   ZERO=Y                                   <- Clear recorder file
/*
```

Figure 138. EREP: Merge and clear the recorder file

Offload Recorder File to a "history file"

and clear the Recorder File

```
// DLBL HISTOD,'VSE.EREP.HISTOD'
// EXTENT SYS009,SYSWK1,1,0,8620,30
// ASSGN SYS009,DISK,VOL=SYSWK1,SHR
// EXEC IFCOFFLD
/*
```

Figure 139. EREP: Offload and clear the recorder file

Some remarks:

- RAS, ERP and EREP ensure that they do not block each other when they write to the recorder file. This may mean that a record gets lost while EREP is cleaning the recorder file (ZERO=Y).
- When the recorder file becomes full, message

  `0T05I RECORDER FILE FULL. RUN EREP`

  is issued to the VSE console (only once). Further recording is suspended (records are discarded) until the recorder file is cleared.
- The recorder file can also be cleared (formatted) with a VSE IPL. When JCL statement **SET RF=CREATE** has been entered before the first // JOB statement is processed in the BG partition, the recorder file gets formatted with the // JOB statement.

z/VSE Virtual Tape Support

The virtual tape function of z/VSE is described in the z/VSE Administration Guide. This chapter describes important aspects of using and working with the Virtual Tape Support available with z/VSE.

In general, the Virtual Tape support included in VSE is intended to be transparent to applications and to provide the customer with the ability to read from or write to a virtual tape in the same way as if it were a physical tape. For technical and performance reasons, the full range of the capabilities of a physical tape has not been implemented.

Before you use the VSE Virtual Tape Support, please check for PTFs related to your particular VSE release. Refer to the VSE Homepage for more information.

Storage Requirement

The I/O Supervisor allocates a 1MB buffer for each virtual tape in the system getvis area of your VSE system. The buffer space is taken preferred out of the System-Getvis 31 area and PFIXed. It is used to buffer the tape data before writing/reading them to/from the Virtual Tape. So before using the "VTAPE START" command make sure that enough PFIXed space is available in the SYSTEM-GETVIS AREA OF YOUR VSE SYSTEM.

If the system is short on pfixed system getvis storage message 1YN1T (Tape Simulator encountered internal error) is displayed in response to the "vtape start" command.

You can check the free PFIXed System-Getvis space of your VSE system by issuing the AR command "MAP SVA".

Defining a Virtual Tape by Job Control Commands

Every job containing a "VTAPE START,UNIT=<cuu> ..." JCC statement should contain an "ON $CANCEL GOTO <label>" statement where the label points to the "VTAPE STOP,UNIT=<cuu>" command. This makes sure that the virtual tape will always be closed at the end of the job - even if the job is cancelled.

Recommendations for the definition of a VSAM virtual tape file

The VSAM file which should contain the tape image for the VSAM virtual tape must be defined before a "VTAPE START" command can be issued. It is recommended to use the skeleton SKVTAPE in the ICCF library 59 for the definition of a VSAM virtual tape file.

Following are some recommendations how to specify the parameters for the definition of a VSAM virtual tape file:

Share Options Settings for VSAM Virtual Tapes:
The Tape Data Handler will only accept single write or multiple read accesses to a VSAM Virtual Tape from a single z/VSE system. Therefore it is recommended to specify Share Option 1 for the creation of a VSAM virtual tape file.***:*** This is also true for a multiple z/VSE system environment. Multiple write access to a single Virtual Tape will cause unpredictable results. Therefore use also Share Option 1 for the definition of a VSAM tape image file in a multiple VSE system environment.

- REUSE Parameter: Specify NOREUSE if you want to write once on the Virtual Tape and only read it afterwards. Specify REUSE if you want to use the Virtual Tape as a work file.
- Record and Control Interval SIZE: it is recommended to specify 32k for the control interval size. Make sure that the record size is at least 10 bytes smaller than the control interval size.

Recommendations for the definition of remote Virtual Tapes

Starting the Virtual Tape with the correct stack number
When the VTAPE function is configured/started for the first time message *1YN1D Tape Simulator encountered internal error* may be issued. This could be caused by non-matching TCP/IP stack definitions:
The TAPESRVR job is started with
// OPTION SYSPARM='00' and will try to connect to TCP/IP stack '00'. The problem occurs if the TCP/IP stack has been started with a different stack number, for example '02'. Therefore no connection can be established and message *1YN1D* is issued.

On the PC, the Virtual Tape server will not show any connection or error messages at all. To solve that problem include/adapt the
// OPTION SYSPARM='nn' statement in your TAPESRVR startup.
You can use skeleton SKVTASTJ in ICCF library 59 to do so.

File name considerations for remote Virtual Tapes

Windows, UNIX or Linux folder names and file names may contain blanks, therefore the FILE parameter must be enclosed in quotes. A quote within filename must be coded as two single quotes, for example:

```
FILE='D:/Frank''s/Virtual Tapes/vt021401.001'
```

Windows, UNIX or Linux can have more than 100 characters in length. Therefore you may specify FILE='filename' twice or even three times. The filename information is concatenated in storage, thus allowing for a file name LENGTH OF 200 OR EVEN 300. the following example is equivalent to the one from above:

```
FILE='D:',FILE='/Frank''s/Virtual Tapes/',FILE='vt021401.001'
```

Windows usually uses back slashes to separate directories (e.g. C:\vtape\tapeimage.aws). Back slashes often causes EBCDIC to ASCII code page translation problems; they are translated into some incorrect characters. Therefore we recommend using forward slashes (e.g. C:/vtape/tapeiamge.aws) with VTAPE, even when using Windows. Forward slashes usually do not cause codepage translation problems.

The Java runtime automatically converts forward slashes into back slashes on Windows. If back slashes are used it can happen that the filename on Windows is treated as relative path instead of an absolute path and the tape image is created in the Virtual Tape Server's installation directory. This is because Windows does not recognize the path as absolute if the back slashes are translated into some incorrect characters.

Sample jobs

The following sample job illustrates the use of VTAPE to backup a VSE library:

```
// JOB BACKUP (Backup a Library to a PC File)
// ON $CANCEL OR $ABEND GOTO VTAPSTOP
VTAPE START,UNIT=480,LOC=9.164.186.20:2285, C
FILE='D:/VSE Backup/prd2.001'
MTC REW,480
// EXEC LIBR
BACKUP LIB=PRD2 TAPE=480
/*
/. VTAPSTOP
VTAPE STOP,UNIT=480
/&
```

Figure 140. VTAPE backup sample job

The following job illustrates the use of VTAPE in combination with logical units :

```
* $$ JOB JNM=VSAMBKUP,DISP=L,CLASS=0
// JOB VSAMBKUP
// ON $CANCEL OR $ABEND GOTO VTAPSTOP
* THIS JOB BACKS UP VSAM DATASETS
// DLBL IJSYSUC,' catalog ',,VSAM
VTAPE START,UNIT=181,LOC=ip-addr,FILE='filename on workstation',SCRATCH
// ASSGN SYS005,181
// EXEC IDCAMS,SIZE=AUTO
BACKUP ( cluster ) -
REW -
NOCOMPACT -
BUFFERS(3)
/*
/. VTAPSTOP
// ASSGN SYS005,UA
VTAPE STOP,UNIT=181
/&
* $$ EOJ
```

Figure 141. VTAPE backup sample job using logical units

Note: If you omit the ASSGN SYS005,UA, you will receive message 1YN5D TAPE 181 is assigned at vtape stop.

Terminating Virtual Tape operations

Cancellation of an I/O operation to a virtual tape

A CANCEL cuu, FORCE command to a Virtual Tape will not always terminate an I/O operation to a Virtual Tape. But in case of a VSAM virtual tape this is the only possibility to terminate an ongoing I/O operation. Check after the successful cancellation with LISTIO, <cuu> for remaining logical assignments to the virtual tape and un-assign them with the ASSGN SYSxxx,UA command before you reuse the virtual tape.

Update available with DY45848 / UD52135

If you issue a "CANCEL <cuu>,FORCE" command and the CUU is used to access a remote virtual tape, then as consequence of the cancel command it may happen that the CUU can no longer be used to access the virtual tape. To be able to re-use the CUU again, do the following:

- Issue a "VTAPE STOP" command for the CUU whose I/O was canceled.
- Define the virtual tape again with a "VTAPE START ..." command.

In case of a remote virtual tape, the easiest way to cancel an ongoing I/O operation is to flush the connection:
Enter

```
xxx FLUSH <ip-address of the remote vtape>,<port of remote vtape>
```

where xxx is the partition ID of the TCP/IP partition.

This causes TCP/IP for z/VSE to flush all data and terminate all connections with the specified IP and port address. The virtual tape can no longer be used after the flush command. After the flush command, check for any logical assignment to the virtual tape by entering the AR command: LISTIO <cuu> where <cuu> is the

physical address of your virtual tape. Issue an ASSGN, SYSxxx,UA command for each assignment that still exits for the virtual tape. After that terminate the virtual tape with:

```
VTAPE STOP,UNIT=<cuu>.
```

This will cause the necessary cleanup so that you should be able to restart the virtual tape with a VTAPE START command after that.

Termination of the Tape Data Handler

The Tape Data Handler is to be considered as a subsystem in the VSE system. Therefore it is not recommended to cancel the Tape Data Handler with the intention to terminate virtual tape processing.

If you want to terminate the Tape Data Handler Subsystem then close all virtual tapes with the "VTAPE STOP" command. The Tape Data Handler will then automatically terminate after 30 seconds. To find out which tape units are defined as virtual, use the VOLUME command. After the termination of the Tape Data Handler verify after that no virtual tapes are active any more. In case there are still virtual tapes defined, issue a "VTAPE STOP against the remaining virtual tapes.

In case the Tape Data Handler is canceled due to an error condition (for instance program check) you must close all open virtual tapes to avoid inconsistencies in the system. Display the open virtual tapes with the AR command "VOLUME". A tape device type of "VTAP-00" in the command output indicates a virtual tape.

Performance considerations

If there is a remote Virtual Tape started whose TCP/IP connection is slow, then this may have an impact on the performance of other Virtual Tapes. Therefore ensure that the TCP/IP connection to the Virtual Tape is stable and fast enough to handle high data transfer rates.

Using TCP/IP, the performance of sending data to a remote host can be improved be setting the "Send Performance Option". This option is by default activated in $SOCKOPT.PHASE. The "Send Performance Option results in TCP/IP sending out data packets without waiting for an ACK (acknowledgement).

The following job provides a $SOCKOPT.PHASE with the "Send Performance Option set (BSDCFG1=$OPTSNWT). Ensure, that the library with the $SOCKOPT.PHASE is in front of the TCP/IP *PRD2.TCPIPC* library in the LIBDEF SEARCH CHAIN.

```
* $$ JOB JNM=SOCKOPT,CLASS=0,DISP=D
// JOB $SOCKOPT
// OPTION CATAL
LOG
// LIBDEF *,SEARCH=PRD2.TCPBETA
// LIBDEF *,CATALOG=PRD2.TEST
* Catalog a new $SOCKOPT phase for TCP/IP 1.5E which includes the
* CSI-supplied options plus the "do not wait after send" option.
* Please ensure that the library with this new $SOCKOPT phase is in
* front of your TCP/IP library within your appl's LIBDEF chain.
NOLOG
// PAUSE OK? Press ENTER to continue
// EXEC ASMA90,SIZE=ASMA90,PARM='SZ(MAX-200K,ABOVE)'
PUNCH ' PHASE $SOCKOPT,* '
$SOCKOPT CSECT
*
* * This phase is used by the BSD-C and SSL interfaces to set options
* * that affect the operation of the interface in a partition...
*
         SOCKOPT  CSECT,                   Generate a csect
               SYSID=00, TCP/IP sysid when BSDCFG1 contains $OPTUSYD
               BSDCFG1=$OPTMECB+$OPTSNWT+$OPTNBSD, options flag 1
               BSDCFG2=$OPTGTSP+$OPTCHKR,  Options flag 2
               BSDXPHS=IPNRBSDC,           Name of BSD/C phase
               CHKT=60,                    Check sockets seconds
               CLST=30,                    Close timeout seconds
               CSRT=30,                    Socket reuse seconds
               QUEDMAX=3,                  Max queued connects allow
               RCVT=00,                    Receive timout seconds
               SNOWMAX=262144,             256K max for send nowait
               SSLCIPH=A,                  SSL default cipher suites
               SSLDEBG=00,                 SSL debugging flag
               SSLFLG1=00, $OPTSRQC,       SSL req close_notify aler
               SSLFLG2=00, $OPTSFZA,       Force use Z/arch instruct
               SSLSTOR=80                  SSL storage usage percen
*
         END   $SOCKOPT
/*
/&
* $$ EOJ
```

Figure 142. Sample SOCKOPT phase

Restrictions

Limitation for writing data to a VSAM virtual tape

The VSAM virtual tape is intended for sequential read and write access. After a "VTAPE START" command data can be written sequentially to the VSAM virtual tape starting from the beginning of the tape. No partial overwrite of data is possible. It is only possible to overwrite all the data starting from the beginning of the tape. To do this, position the tape at the beginning and then write your data.

Make sure, that the virtual tape is defined with the "REUSE" parameter, otherwise no overwrite at all is possible. Once a "VTAPE STOP" command has been issued the EOV ("end of volume") indication is written to the virtual tape and no further data can be appended. Even when the virtual tape is re-defined with a new

"VTAPE START" command, writing to the tape must start again at the beginning of the tape.

Stacking of multiple BACKUPs , like LIBR Backup can lead to a PERM error with sense data

```
196  write rejected due to internal respositioning at EOF.
```

No alternate tape support:

There is no alternate tape support for virtual tapes. In case of a VSAM virtual tape choose the primary and secondary allocation of the virtual tape file large enough so that an "end of volume" condition does not occur during processing of the virtual tape. In case of a remote virtual tape the size of the remote virtual tape file is assigned due to the PC file rules. Look at them if you get an "end of volume" condition for your remote virtual tape.

The DITTO "ERASE TAPE" function is not supported for virtual

tapes: An application issuing the ERASE TAPE function will be cancelled. Use the equivalent VSAM and PC file system functions instead.

SDAID does not support virtual tapes to be used as OUTDEVICE

SDAID does not support virtual tapes to be used as OUTDEVICE. However, tracing virtual tape I/Os is possible.

No virtual tape support in a stand alone environment

There is no virtual tape support in a stand alone environment.

Current restriction with respect to label processing

The following job sequence will cause a deadlock between the BG and the Tape Data Handler partition:

```
// OPTION STDLABEL=ADD
// DLBL VTAPFIL,'VTAPFIL.TEST.FILE',,VSAM,CAT=IJSYSCT
// VTAPE START,UNIT=181,LOC=VSAM,FILE='VTAPFIL'
```

The problem is that the VTAPE command will cause a deadlock, if the BG partition is concurrently updating system standard labels (that is // OPTION STDLABEL is in effect).

Circumvention: Terminate the standard label processing for example by issuing "// OPTION USRLABEL" before the 'VTAPE START' statement. A correct job control sequence is therefore:

```
// OPTION STDLABEL=ADD
// DLBL VTAPFIL,'VTAPFIL.TEST.FILE',,VSAM,CAT=IJSYSCT
// OPTION USRLABEL
// VTAPE START,UNIT=181,LOC=VSAM,FILE='VTAPFIL'
```

Diagnostic Aids

In case of an error situation in virtual tape processing, z/VSE will sometimes return sense information for the Virtual Tape. The sense information contains a reason code in the bytes 14-15 and a return code in bytes 16-17.

Example:

```
BG 14 0P49I C PERM ERROR SYS_99=280
      CCSW=010051D1B0020040DC CCB=51D178
      SNS= 10442047 00002120 00000000 00000196 00C80088 04010101
           010100F0 000000FF
```

In the above example the return code is x 00C8 and the reason code is x 0196

The Return and Reason Codes are also contained in the Trace for the Tape Data Handler.

Return Codes (hex):

```
0064 TCP/IP connection error
00C8 VTAPE error
0111 supervisor encountered error
012C internal error
0190 Tape data handler was canceled
```

Reason Codes (hex):

```
0067 allocation of control blocks failed
0069 write access to vtape requested but only read access allowed
006F host name could not be resolved
00CD the same virtual tape is used concurrently with read and write
access
00CE user requested cancel
00CF Tape Data Handler canceled
012F no socket received for connection
0130 session could not be established - maybe wrong ip address
specified
0132 unrecoverable send error
0133 unrecoverable read error
0134 error when closing the connection
0135 allocation of control blocks failed
0138 timeout when establishing the session - no session established
0191 invalid parameter
0192 null pointer , coding error or GETVIS problem
0193 OPEN failure of VTAPE image
0195 no valid AWSTAPE format
0196 write rejected , not EOF or file not defined as REUSE
0197 read error , e.g I/O error
0198 write error e.g no more space / extents available
019A repositioning error , internal error e.g I/O error
019C buffer handling error , invalid buffer format
019E buffer too small, internal error
0222 canceled by MIH , internal error
0314 failed to establish connection to remote host
0333 tape data handler not active / canceled
0444 excessive VTAPE usage
```

Tracing the Tape Data Handler

To enable the VTAPE trace, change the TAPESRVR job (use skeleton SKVTASTJ in ICCF lib 59)

// EXEC $VTMAIN,SIZE=$VTMAIN,PARM='TRACE'

and set DEBUG ON. The Trace messages will go to SYSLST (job output).

Note: The TAPESRVR completes with a return code of 0008 when running with trace active.

Tracing remote Virtual Tapes

To trace the Virtual Tape Server running on a java platform do the following:

1. Set messages=on in the VirtualTapeServer.properties file. This will show some more trace messages written to STDOUT. See especially the Exceptions in this trace.
2. Edit the run.bat or run.cmd start script and change the java call as follows: java -Dcom.ibm.vse.vtape.trace=trace.txt com.ibm.vse.vtape.VirtualTapeServer This activates the internal trace. Trace output goes to a file named trace.txt in the current directory.
3. Set the TCP/IP BSD Trace on. Rename XSOCKDBG.PHASE in PRD2.TCPIPC to $SOCJDBG.PHASE. This shows Trace messages on SYSLOG/SYSLST for each Socket call.

Browser for AWSTAPE files

To aid those users seeking tools and information for so-called AWS or AWSTAPE virtual tapes, the following links may be of help:

http://www.cbttape.org/awstape.htm or
http://www.cbttape.org/ fish/hercgui-index.html

ListVOL1 utility

The ListVOL1 utility (download here
http://www.ibm.com/systems/z/os/zvse/downloads/tools.html#listvol1
reads the first 2 tape records of tape image in AWSTAPE format residing in a VSA M ESDS cluster. It prints the VOLSER and FILEID from the VOL1 and HDR1 labels on the tape. ListVOL1 runs in a VSE partition (dynamic or static). The partition size must be at least 5 MB. ListVOL1 is started using

```
// EXEC LISTVOL1
<DLBL name of VSAM file>
<DLBL name of VSAM file>
/*
```

where <DLBL name of VSAM file> specifies the DLBL name of a VSAMESDS cluster containing a VTAPE tape image in AWSTAPE format.

The output looks as follows:

```
LISTVOL1 UTILITY - LIST VOL1/HDR1 LABELS OF VTAPES
FILENAME: VOLSER FILE-ID
-------------------------------------------------
VTAPE1 : TAPE TAPE.DATASET.
VTAPE2 : PRDDAT PRODUCTON.DATA
VTAPE3 : BACKUP MY.BACKUP.FILE
-------------------------------------------------
LISTVOL1 UTILITY - FINISHED
```

Messages 1YM7T or 1YN1D Errors

The following error messages 1YM7T TAPE DATA HANDLER ENCOUNTERED CONNECTION ERROR or 1YN1D TAPE SIMULATOR ENCOUNTERED INTERNAL ERROR may have other explanations than found in the message explanation:

1. TCP/IP for z/VSE has been started with a system whose system ID is specified:

   ```
   // EXEC IPNET,SIZE=IPNET,PARM='ID=nn,INIT=.....
   ```

   Where the default for "nn" is "00". To use the TCP/IP services from within another partition this partition has to 'know' the system ID. This is specified as follows:

   ```
   // OPTION SYSPARM='nn'
   ```

   where 'nn' is the system ID. Customers may use the skeleton SKVTASTJ in ICCF lib 59 to add this statement. This is described in TCP/IP for VSE Programmer's Reference

2. The wrong $EDCTCPV.PHASE is used. VSE ships a dummy $EDCTCPV.PHASE in PRD2.SCEEBASE. TCP/IP ships a second $EDCTCPV.PHASE (typically in PRD2.TCPIPC or in the product library). In case the dummy phase from PRD2.SCEEBASE is used, socket calls will not work. Instead an EDCxxx message is issued to SYSLST: "EDCxxx Function nnn not implemented". Make sure the $EDCTCPV.PHASE from TCP/IP is used (adjust the LIBDEFs).

VTAPE doesn't use PC directory specified by FILE parameter

When specifying file names, the following should be considered.

- Windows, UNIX or Linux folder names and file names may contain blanks, therefore the FILE parameter must be enclosed in quotes. A quote within filename must be coded as two single quotes, for example:

  ```
  FILE='D:/Frank''s/Virtual Tapes/vt021401.001'
  ```

- Windows, UNIX or Linux can have more than 100 characters in length. Therefore you may specify FILE='filename' twice or even three times. The filename information is concatenated in storage, thus allowing for a file name length of 200 or even 300. The following example is equivalent to the one from above:

  ```
  FILE='D:',FILE='/Frank''s/Virtual Tapes/',FILE='vt021401.001'
  ```

- Windows usually uses back slashes to separate directories (e.g. C:\vtape\tapeimge.aws). Back slashes often causes EBCDIC to ASCII code page translation problems; they are translated into some incorrect characters. Therefore we recommend using forward slashes (e.g. C:/vtape/tapeiamge.aws) with VTAPE, even when using Windows. Forward slashes usually do not cause codepage translation problems. The Java runtime automatically converts

forward slashes into back slashes on Windows. If back slashes are used it can happen that the filename on Windows is treated as relative path instead of an absolute path and the tape image is created in the Virtual Tape Server's installation directory. This is because Windows does not recognize the path as absolute if the back slashes are translated into some incorrect characters.

Problems accessing real tape

If you have problems acessing a real tape, you should check with the QT command, if a virtual tape using the same adress is still active.
It is possible that the virtual tape definition has been started, but was never stopped.

Chapter 24. Support of 4-digit physical device addresses

Introduction

4-digit device addresses mean device addresses larger than x"FFF".

The support of 4-digit device addresses removes constraints in the I/O area. In special it enables z/VSE to recognize and to work with devices which have device addresses larger than x"FFF".

I/O devices can have physical device numbers in the range of x"0" to x"FFFF" . But prior to z/VSE 4.3 only I/O devices with device numbers in the range of x"0" to x"FFF" were supported. I/O devices with device numbers > x"FFF" were ignored during the z/VSE installation process and could not be added in the IPL procedure.

z/VSE 4.3 and later relases support I/O devices with device numbers in the range of x"0" to x"FFFF". I/O devices with device numbers > x"FFF" were recognized during the z/VSE installation process and can be added into the IPL procedure.

In the past the system administrator always had to make sure that devices assigned to z/VSE had a physical device number equal or less than x"FFF". Now, he does not have to care any more for this. He can assign any devices to z/VSE regardless of their physical device number.

Physical device addresses and VSE addresses

Starting with z/VSE 4.3, a so called "VSE address" is assigned to each device. The VSE address is always in the range of x"0" to x"FFF".

If the physical device address of an I/O device is <= x"FFF" then - by default - the VSE address is equal to the physical device address.

If the physical device address of an I/O device is > x"FFF" then a VSE address in the range between x"0" and x"FFF" will be assigned to the device.

z/VSE internally only works with the VSE address of the device.

Assigning a VSE address

The assignment of a VSE address to a physical device address is done by using the IPL ADD statement. The IPL ADD statement was extended for this. Here is the new syntax of the IPL ADD statement:

Syntax:

ADD <pcuu > as <cuu>,<device type>

ADD <pcuu1> : <pcuu2> as <cuu1> : <cuu2>, <device type>

ADD <pcuu1> .. <pcuu2> as <cuu1> .. <cuu2>, <device type>

where
<pcuu> is the physical device address,
<pcuu1> is the low physical device address and
<pcuu2> is the high physical device address.
<cuu> is the VSE address,
<cuu1> is the low VSE address and
<cuu2> is the high VSE address.

You only have to specify a VSE address if your physical device address is larger than x"FFF" as otherwise the VSE address is equal to the physical device address by default.

The VSE address must be unique: You cannot assign the same VSE address twice and you cannot specify a VSE address which matches with an existing physical device address.

This assignment between physical device addresses and VSE addresses cannot be changed any more after IPL.

Examples:

Assume your IPL procedure contains the following ADD statements:

```
ADD 380,ECKD
ADD 200,ECKD
ADD 22A3 as FFA,TPA
ADD FA04 as A04,TPA
```

Then the assignment of physical device addresses to VSE addresses would be as follows:

```
Physical device address VSE address
 380                          380
 200                          200
 22A3                         FFA
 FA04                         A04
```

The following ADD statements are not allowed:

```
ADD  300 as 500     NOT allowed
ADD 3A05 as 200     NOT allowed if a device with the physical
                                 device address 200 already exists
```

Addressing a device

We have discussed so far that you have both: a physical device address and a VSE address for each device. Now let's come to the question when you have to use the physical device address and when you have to use the VSE address.

If you interact with the hardware directly or if you interact with VM then you must use the physical device address as - neither the hardware nor VM - know the VSE address.

But if you interact with z/VSE, you have to use the VSE address and vice versa. z/VSE will show you the VSE address if messages were displayed or in logs.

Example:

Let us assume you have 4 DASDs with the physical device addresses 8000-8003 and you want to use these devices with z/VSE.

You have to define these DASD in your IOCP hardware definition by using the physical device addresses 8000 to 8003.

```
IOCP:
CNTLUNIT CUNUMBR=8000,PATH=(A1,B1,A2,B2,A3,B3,A4,B4),
IODEVICE ADDRESS=(8000,4),CUNUMBR=(8000),FEATURE=(SHARED),UNIT=3390B
```

On the VSE side, you have to ADD these devices in the IPL procedure and assign them VSE addresses.

```
ADD 8000:8003 as 800:803,eckd
```

When you want to access one of the DASDs, you could assign them to a logical unit for example by using the VSE address in the ASSGN JCL stmt.

```
// JOB CLRDK
// ASSGN SYS012,801
// DLBL UOUT,'DISK.XXXXX',9999
// EXTENT SYS012,,,,2590,5
// EXEC CLRDK
// END
/&
/*
```

Limitations

Limitations mean: there are some areas within VSE where you cannot or do not want to hide the physical device address. This mainly applies to the early phases of IPL when the ADD stmt which assigns a VSE address is not yet processed.

When running z/VSE you may see the physical device address

- in error messages at *early* IPL time
- in the Supervisor Control Statement when specifying the console address.

  When processing the Supervisor Control Statement, the assignment of a physical device address to a VSE addresses is not already known as the assignment is done in the ADD statement which follows the Supervisor Control Statement.

  Therefore you have to specify the physical device address in the Supervisor Control Statement to address the console. For example:

  ```
  A009,$$A$SUPI,VSIZE=264M,VIO=512K,VPOOL=64K,LOG,IODEV=1024
  ```

  The dialog *Tailor IPL procedure* also requires the physical device address of the console.
- in command output showing alias devices (PAV Support)

  A PAV device consists of a base device and one or more alias devices. The z/VSE user can only address the PAV base device. Therefore there is no need to assign VSE addresses to alias devices. That's why you see the alias devices with their physical device address.
- In the Recorder File

The Recorder File contains the physical device address because the Recorder File is the interface to the hardware people and they have to know the real address of the device.

Installation with 4-digit physical device addresses

During the installation process several times the user has to enter device addresses. At initial installation time this is in general the *physical device address*.

The system internally assigns VSE addresses to all devices recognized.

If the physical device address is higher than FFF, messages like the following ones list both physical and VSE address.

```
SA07I THE PHYSICAL DEVICE ADDRESS C001 CORRESPONDS TO THE ADDRESS
         101 USED BY VSE

SI18I DOSRES PHYSICAL DEVICE ADDRESS IS A300, VSE ADDRESS IS
                      008, DEVICE TYPE ECKD
```

Be aware that the mapping for a device is valid during the actual IPL.

The only exception is ICKDSF, the ICKDSF utility requires the VSE address.

```
SA60I ***** FOR THE ICKDSF UTILITY YOU NEED TO SPECIFY VSE DEVICE
    ADDRESSES WHICH YOU CAN FIND BY USING THE QUERY IO COMMAND. *****
```

More than 1024 devices

If more than 1024 devices were recognized during the installation process then all devices recognized are displayed on the console using the IPL DEV command. The DEV command shows the physical device address of the devices. And the user has to specify the physical device address when issuing the DEL command to delete devices.

Example:

```
0 dev
BG 0000 DEVICES ADDED AND/OR SENSED:
BG 0000  CUU RANGE   DEVICE TYPE
BG 0000   0009          3277
BG 0000   000C          2540R
BG 0000   000D          2540P
BG 0000   000E          1403
BG 0000   0150:0151     ECKD
BG 0000   0180:0181     ECKD
BG 0000   0190:0194     ECKD
BG 0000   019B          ECKD
BG 0000   019D:019E     ECKD
BG 0000   0200:0201     ECKD
BG 0000   0280:0281     3480
BG 0000   2000          ECKD
BG-0000
0 DEL 2000
```

```
SYSTEM:  z/VSE              z/VSE 5.1         TURBO (01)       USER:
VM USER ID: VSE150                                             TIME:
BG 0000 DEF SYSCAT=DOSRES,SYSREC=SYSWK1
BG 0000 0J99I  DEVICE ADDRESS    VSE ADDRESS ASSIGNED
BG 0000             B380           001
BG 0000             B381           002
BG 0000             B382           003
BG 0000             C380           004
BG 0000             C381           005
BG 0000             C382           006
BG 0000             E380           007
BG 0000             E381           008
BG 0000             E382           00A
BG 0000             F150           00B
BG 0000             F151           010
BG 0000             F480           011
BG 0000             FF80           012
BG 0000             FF81           013
BG 0000             FF82           014
BG 0000 SYS DASDFP=YES
BG 0000 SYS JA=YES
BG 0000 SYS SPSIZE=0K
==>

1=HLP 2=CPY 3=END       6=CNCL 7=BWD 8=FWD 9=EXPL 10=INP 11=PCUU 12=RTRV
```

First time use

When the dialog Configure Hardware is entered for the first time, the Unidentified Device List is shown.

On this panel all devices are listed, which

- Could not be sensed completely
- have a physical address higher than FFF and therefor have a VSE mapping.

This panel shows the 4-digit physical device address and the VSE address which was internally assigned to the device. You can assign a different VSE address to the device by using the option 6 (alter mapping).

Later the VSE address of a device can not be changed any more, but it must be deleted and added with a new definition.

```
ADM$HDWF      HARDWARE CONFIGURATION: UNIDENTIFIED DEVICE LIST


OPTIONS:  1 = DEFINE A DEVICE    5 = DELETE A DEVICE
          6 = ALTER MAPPING
         '=' = REPEAT LAST DEFINED DEVICE

   OPT     VSE    PHYSICAL  DEVICE  DEVICE TYPE  DEVICE   SPECIFICATION
           ADDR    ADDR               CODE          (MODE)
    _      001     B380     ?          3277
    _      002     B381     ?          3277
    _      003     B382     ?          3277
    _      004     C380     ?          3277
    _      005     C381     ?          3277
    _      006     C382     ?          3277
    _      007     E380     ?          3277
    _      008     E381     ?          3277
    _      009     0009     ?          3277
    _      00A     E382     ?          3277
PF1=HELP       2=REDISPLAY  3=END                    5=PROCESS
               8=FORWARD
```

Useful commands and dialogs

Displaying the VSE address / the physical address

There is a new AR cmd which allows you to display the assignment of physical device addresses to VSE addresses.

```
QUERY  IO-------------------------------------------------------------
                  |--,CUU=-----------------------------------------|
                                    |all------------|
                                    |cuu------------|
```

where cuu can be either a physical device address or a VSE address. If you specify a physical device address then you get the VSE address as result.

And if you specify a VSE address then you get the physical device address as result.

Examples:

```
QUERY IO,CUU=1200
AR 0015  PHYSICAL ADDRESS    ADDRESS USED BY Z/VSE   DEVICE CLASS
AR 0015        1200                  200                    DASD

QUERY IO,CUU=FFB
AR 0015  VSE ADDR   PHYSICAL ADDR   DEVICE CLASS
AR 0015        FFB      2000            DASD
```

PF11 PCUU at the console

On the VSE console the new key PF11=PCUU is offered. When the cursor is positioned at a VSE address and PF11 is pressed, the required QUERY IO command to retrieve the physical address is displayed on the command line. You only have to press enter to get the assignment.

Print configuration list

The selection *print configuration list* offered via PF9 Print in the Hardware Configuration Dialog now prints an overview of the physical device addresses and their mapped VSE addresses.

GETFLD Macro Extension

The internal GETFLD macro is extended to return the physical and the VSE address of a device. The syntax is as follows:

```
GETFLD FIELD=CUUMAP,CUU={name|(Rx)|(0)}
```

where cuu specifies either the physical device address or the VSE address.

It returns both the physical device address in register 0 and the VSE address in register 1.

Chapter 25. z/VSE Initial Installation using an Installation Disk

Introduction

Starting with z/VSE 5.2, z/VSE supports initial installation from an installation disk for both the LPAR and z/VM guest environment. Initial installation from a physical tape is still supported.

Be aware that

- The installation disk can be used for initial installation only.
- Fast Service Upgrade, the installation of optional products and the installation of the generation feature still require a physical or a virtual tape.

z/VSE provides utilities to create an installation disk both in an LPAR and z/VM CMS environment.

How to create an installation disk and how to perform initial installation using an installation disk is described in the z/VSE Installation manual.

Note: In the following text these variables are used
YY defines the LANGUAGE of the z/VSE base, in which the system was ordered:
YY = EN for the English version.
YY = KA for the Kanji version.
vrm defines the version, release and modification level of z/VSE.
Be aware that starting with z/VSE V6.1, z/VSE is delivered as English version only.

Prerequisites:

- The installation disk must be a 3390 type ECKD disk with at least 500 cylinders.
- To create the installation disk in an LPAR environment, z/VSE requires at least 512MB of processor storage.
- Starting with z/VSE 5.2, z/VSE requires at least 64MB of processor storage (both z/VM and LPAR environment).
- To IPL the installation disk, z/VSE requires at least 64MB of processor storage (both z/VM and LPAR environment).
- The version of the utilities you are using to create the installation disk must match the z/VSE base tape image (AWS file):
  - Always use the utilities shipped with your order of z/VSE to create the installation disk.
  - Older versions of the utilities must not be used. They might create an invalid installation disk.

How to get the utilities to create the installation disk

The utilities are shipped with your order of z/VSE.

- If you order a DVD, the utilities are located in the root directory of your DVD.
- If you choose internet delivery, please download and unzip the file containing the utilities. Details can be found in the z/VSE Installation manual.

Initial installation:

IPL pcuu (pcuu is physical device address of the installation disk).

Proceed as described in the z/VSE Installation manual.

Problem determination

In case a problem occurs during initial installation:

In which environment was the installation disk created? LPAR or z/VM? Was the correct version of the utilities used?

If the installation disk was created correctly, please contact IBM support. Only if requested by IBM support, IPL the installation disk in debug mode:

```
z/VM:   IPL pcuu LOADPARM ....D
LPAR:   Set ....D in the LOAD PARAMETER field
```

When IPLing with debug enabled the system displays debug messages, for example

```
BG 0000 DEBUG: In VTCTDH
BG 0000 DEBUG: Call EDCXHOTC
 ..
BG 0000 DEBUG: AWSTapeOpen called
BG 0000 DEBUG:  szFileName  = CUU=202,FILE=ZVSE920.BASE.TAPE.AWS
```

When the system stops processing, please send the latest messages that were displayed to IBM support.

Note: You can set *%SET PAUSE 00* to scroll messages faster. An initial installation with debug enabled can take two hours or more.

In case a problem occurs when creating the installation disk:

z/VM CMS environment:

The z/VM utility (VSEIDISK) will display error messages with prefix IDSK. If it is not a setup / configuration problem (for example invalid syntax, disk too small ...) please send the error messages to IBM support.

LPAR environment:

The LPAR utility will display error messages with prefix IDSK. When the system stops processing please send the error messages to IBM support. If the error messages are not sufficient to solve the problem, IBM may advise you to create the installation disk with debug enabled. The system will then display detailed error messages. To create the installation disk with debug enabled, please select the VSEvrmxxD.INS file, or when the integrated console is used, VSEvrmxxID.INS.

The error messages with prefix IDSK are described in the Messages and Code manual and in the OME.

When the installation disk was created the system displays message

```
BG 0000 SA17W ***** END OF STAND ALONE PROCESSING *****
```

and enters hardwait. This is not an error. You can now IPL the installation disk to start initial installation.

Pitfalls

These are pitfalls that can be avoided:

- Customers who experience message ACTZ0197 when creating an installation disk in an LPAR could check:
  - Does the LPAR profile specify at least 512MB processor storage?
    If yes, was the LPAR deactivated / activated to activate any changes in the profile?
  - Are both the HMC and SE allowed to access the FTP source?
  - Are all required files on the DVD or in the FTP directory, that is both the utilities and the z/VSE base tape image file VSEvrmEN.AWS?
  - Is the name of the z/VSE base image file VSEvrmEN.AWS?
  - Does the version of the z/VSE base image file (vrm) match the version of the file VSEvrmEN.INS (or any other INS file)?

  For a list of all required files, see the z/VSE Installation manual.
- The z/VM guest requires at least 64 MB storage. Otherwise message HCPVMI232E IPL UNIT ERROR will occur when using the VSEIDISK utility to create the installation disk.
- Customer who experience message SI83I (likely with return code 0195) followed by message 0P32I during installation of z/VSE from installation disk should verify whether the z/VSE installation tape image was correctly transferred in binary transfer mode without CRLF end-of-record processing to variable-length z/VM CMS file. When using 3270 file transfer (IND$FILE) the option to treat CRLF as end-of-record delimiter must be disabled otherwise the tape image is corrupted. After having retransferred the tape image with the correct options the installation disk must be recreated.

Index

Special Characters

Numerics

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

Z

Communicating Your Comments to IBM

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply.

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of the book. However, the comments you send should pertain to only the information in this manual and the way in which the information is presented. To ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

- You can send comments directly from the z/VSE home page. Go to:

  http://www.ibm.com/systems/z/os/zvse/

- If you prefer to send comments electronically, use this Internet address:

  s390id@de.ibm.com

- If you prefer to send comments by FAX, use this number:

  (Germany): 07031+16-2636
  (Other countries): (+49)+7031-16-2636

Make sure to include the following in your note:

- Title and edition of this book
- Page number or topic to which your comment applies.

XX00-0000-00