

Virtual PARS

VPARS

User's Guide and Reference

Version 4 Release 2 Level 00



Virtual Software
Systems, Inc.

Preface

Virtual PARS 4.2 at service level 00 and above on 08/02/03.

Copyright Virtual Software Systems, Inc., 1985-2003.

Copies of this publication may be printed for the internal
use of licensed or trial users of Virtual PARS.
This notice must appear in all copies.

Virtual Software Systems, Inc.
7715 Browns Bridge Road
Gainesville, Georgia 30506
(770) 781-3200
www.vsoftsys.com

Extended Architecture, VM/XA, Enterprise Systems Architecture, VM/ESA,
z/VM and IBM are trademarks of International Business Machines Corporation.

Table of Contents

Year 2000 compliance

Chapter 1. Year 2000 compliance	1
Description.	1
VM releases supported.	1
Date format supported.	1
Installation default.	1
VSSI Date commands.	2
Principle of operation.	2
Online code.	2
CMS code.	2

VSSI Commands

Chapter 2. VSSI Commands	3
VSQUERY	4
VSQUERY DATEFORMAT	5
VSQUERY VSIDISK	6
VSQUERY VSLEVEL	7
VSSET	8
VSSET DATEFORMAT	9
VSSET FULLDATE	10
VSSET SHORTDATE	11
VSSET ISODATE	12
VSSET SYSDATE	13
VSSET DELIMITER	14
VSSET VSIDISK	15

VPARS User Information

Chapter 3. Introduction and Concepts	17
Backups and Restores	18
Chapter 4. Using VPARS	19
TPF Database Processing	19
VPARS Database Configurations	19
Multi-Level Databases	20
Loosely-Coupled Testing	20
Testing Without a TPF Base System	20
PTV Operation	20
Description	21
Method of Operation	21
Testing IPL Text	21

Deleting Records	21
Chapter 5. VPARS Command Summaries	23
CP Commands	23
CMS Commands	24
Chapter 6. VPARS CP Commands	27
VPACC	28
VPADD	29
VPCLEAR	30
VPCLOSE	32
VPFCLOSE	33
VPINIT	34
VPIPL	35
VPLINK	36
VPOPEN	38
VPQUERY	42
VPQUERY ACTIVITY	43
VPQUERY BUFFERS	45
VPQUERY CONCAT	47
VPQUERY CONFIG	48
VPQUERY DEFAULTS	50
VPQUERY DEVTBL	52
VPQUERY FMTTBL	53
VPQUERY KEY	55
VPQUERY LOCKQUEUE	57
VPQUERY MAXUSERS	59
VPQUERY MDISKPOOL	60
VPQUERY OPEN	61
VPQUERY STATUS	63
VPQUERY TPFSTATUS	65
VPQUERY USERS	67
VPQUERY VPSTATUS	69
VPREL	71
VPREMOVE	72
VPSET	73
VPSET BASE	74
VPSET BUILD	75
VPSET CHECKPOINT	77
VPSET CLEAR	78
VPSET COUPLED	79
VPSET ECHO	80
VPSET INTERCEPT	81
VPSET MAXONQ	82
VPSET MAXUSERS	83
VPSET MINONQ	84
VPSET PTV	85
VPSET REQONLINE	86
VPSET STATUSTIME	87
VPSET UNSUPCCW	88
VPSET UNSUPMSG	89

VPTEST	90
Chapter 7. VPARS CMS Commands	91
VPBKUP	92
VPBLDFMT	96
VPBXMAMP	97
VPBXPLTB	99
VPBXREST	100
VPFMT	103
VPIPLWR	105
VPLOAD	107
VPREST	110
VPUNLD	113
VPUTIL	116
Chapter 8. VPARS CMS Execs	119
VPBXPLTB	120
VSSLTAPE	121

VPARS System Programmer Information

Chapter 9. Planning for VPARS	123
VPARS Database Description	123
Database Device Numbers	123
Database Minidisk Allocation	124
Database Minidisk Sizes	124
Multi-Level Databases	125
Loosely-Coupled Testing	125
The TPF Base System	126
TPF Record Sizes	126
Testing Without a TPF Base System	126
Backups and Restores	127
Chapter 10. Customization Overview	129
Chapter 11. Creating Configuration Definitions.	131
The VPARS System Defaults File	131
VPARS Configuration Definitions	134
VPARS Concatenation Definitions	138
TPF Device Table Definitions	139
TPF Format Table Definitions	140
Chapter 12. Post-Installation Customization	143
Chapter 13. VPARS Modifications to VM	145
Chapter 14. PTV Support	147
Description	147
Method of Operation	147

#

Installation	147
PTV Program Modification	147
Chapter 15. Concatenation Tutorial	149
Concatenation Example 1	154
Concatenation Example 2	155
Chapter 16. Directory Requirements	157
Sample Directory Entries	158

VPARS Messages

Chapter 17. VPARS CP Messages	163
Chapter 18. VPARS CMS Messages	195

VPARS Abend codes

Chapter 19. VPARS ABEND codes.	221
--	------------

VSSI Messages

Chapter 20. VSSI CP Messages	225
---	------------

VSSI Abend codes

Chapter 21. VSSI Abend codes	231
---	------------

Chapter 1. Year 2000 compliance

Description.

All VSSI software is YEAR2000 compliant. This means that all VSSI components support a 4 digit year format. The actual format used is defined at installation time and can be modified via CP commands or by configuration file parameters at IPL time.

VM releases supported.

VSSI YEAR2000 compliance is supported in all z/VM releases

Date format supported.

The following date formats are supported by VSSI YEAR2000 compliance. When applicable the corresponding IBM format name is specified in parentheses.

MM/DD/CCYY - (FULLDATE)
DD/MM/CCYY
CCYY/MM/DD - (ISODATE)
CCYY/DD/MM
MM/DD/YY - (SHORTDATE) This is the default at installation.
DD/MM/YY
YY/MM/DD
YY/DD/MM

Installation default.

The installation default is SHORTDATE.

VSSI Date commands.

The following CP commands have been implemented to support the VSSI YEAR2000 compliance: (See the VPARS or VTAPE User's guide for the full command syntax)

- VSSet with the following arguments:
 - VSSet DATEFormat { Anyone of the 8 formats defined above }
 - VSSet { Anyone of the 3 predefined IBM format names }
 - VSSet DELimiter { any character }
- VSQuery with the following argument:
 - VSQuery DATEFormat

Principle of operation.

The default date format is specified during installation by modifying the VSOPTNS Macro.

Online code.

After installation the default date format can be modified by:

- Including the VSI_DATEFormat statement in the VSSI system configuration file. (See the installation guide)
- Issuing the VSSset command with the appropriate parameters.

CMS code.

The default date format, from the VSOPTNS macro, is included in all CMS modules created at installation time. When a CMS module executes, it requests the current date format from the CP using the VSQuery commsnd. If the online date format is available, the CMS module will use it. If the request fails, the CMS module will use the default format.

Chapter 2. VSSI Commands

This section describes the VSSI CP commands. The format, use, and the normal responses of each command are explained. Error responses are not listed here. You can find explanations of error messages in the "Messages" section of this manual, or by using the CMS Help facility.

VSQUERY

The VSQuery command displays various parameters common to all VSSI products. Each subcommand is described separately on the following pages.

VSQuery	Dateformat VSIDisk VSLevel
---------	----------------------------------

VSQUERY DATEFORMAT

Use the VSQuery DAtеformat to display the current format used by VSSI online code.

VSQuery	DAtеformat
---------	------------

Usage Notes:

This command does not accept any parameter.
Since the delimiter is independently modifiable, it is shown as a question mark (?) in the examples below. Only the 'named' date formats imply a fixed delimiter. (ie:FULLDate, ISODate or SHORTDate)

Response:

```
VSSI DATEFORMAT =    FULLDATE      (Format is MM/DD/CCYY)
                    MM?DD?CCYY
                    DD?MM?CCYY
                    CCYY?MM?DD
                    ISODATE      (Format is CCYY-MM-DD)
                    CCYY?DD?MM
                    SHORTDATE    (Format is MM/DD/YY)
                    MM?DD?YY
                    DD?MM?YY
                    YY?MM?DD
                    YY?DD?MM
```

VSQUERY VSIDISK

Use the VSQuery VSIDisk command to display the current Userid/Minidisk definition for the VSSI parameter disk and whether it is accessed.

VSQuery	VSIDisk
---------	---------

Usage Notes:

This command does not accept any parameter.

Response:

```
VSSI Disk at address vdev, owned by userid is not currently accessed.  
VSSI Disk at address vdev, is owned by userid and accessed as c  
There was no definition for the VSSI parm disk in the configuration file  
used at IPL.
```

VSQUERY VSLEVEL

Use the VSQuery VSLevel command to display the current VSSI code level and the VM/ESA level it corresponds to.

VSQuery	VSLevel
---------	---------

Usage Notes:

This command does not accept any parameter.

Response:

```
VSSI level 40.00 for VM/ESA 2.2.0
```

VSSET

The VSSet command sets various parameters common to all VSSI products. Each subcommand is described separately on the following pages.

VSSet	<p>FULLDate ISODate SHORTDate SYSDate DATEformat [<i>options</i>] DELIMiter [<i>c</i>] VSIDisk <i>userid</i> <i>vdev</i> [Select A]</p> <p><i>options</i> MM/DD/CCYY DD/MM/CCYY CCYY/MM/DD CCYY/DD/MM MM/DD/YY DD/MM/YY YY/MM/DD YY/DD/MM</p> <p><i>c</i> Select A Any REPlace char. FOrce</p>
-------	---

VSSSET DATEFORMAT

Use the VSSet DATEformat to set the date format used by VSSI online code.

VSSet	DATEFormat {MM/DD/CCYY} {DD/MM/CCYY} {CCYY/MM/DD} {CCYY/DD/MM} [DELimiter c] {MM/DD/YY } {DD/MM/YY } {YY/MM/DD } {YY/DD/MM }
-------	--

Usage Notes:

The DATEFormat command does not change the delimiter. The delimiter is changed by using:

- the DELimiter command on the same VSSet as the DATEformat
- the DELimiter command on its own VSSet
- a 'named' format. (ie: VSSet ISODATE implies the delimiter -)

Responses:

VSSI DATEFORMAT = {format specified on the command}

VSSet FULLDATE

Use the VSSet FULLDate command to change the date format to MM/DD/CCYY

VSSet	FULLDate
-------	----------

Responses:

VSSI DATEFORMAT = FULLDATE

VSSet SHORTDATE

Use the VSSet SHORTDate command to change the date format to MM/DD/YY

VSSet	SHORTDate
-------	-----------

Responses:

VSSI DATEFORMAT = SHORTDATE

VSSet ISODATE

Use the VSSet ISODate command to change the date format to CCYY-MM-DD

VSSet	ISODate
-------	---------

Responses:

```
VSSI DATEFORMAT = ISODATE
```

VSSSET SYSDATE

Use the VSSet SYSDate command to change the VSSI date format to the IBM date format currently active

VSSet	SYSDate
-------	---------

Responses:

VSSI DATEFORMAT = {FULLDATE or SHORTDATE or ISODATE}

VSSSET DELIMITER

Use the VSSset DELimiter command to change the date delimiter.

VSSset	DElimiter { c }
--------	------------------------

c Any character to be used as the date delimiter.

Responses:

```
VSSI DATEFORMAT = {current date format with new delimiter}
```

VSSet VSIDISK

Use the VSSet VSIDisk command to change the definition (owner and virtual address) of the VSSI parm disk. This information is usually set at IPL time, by specifying it on the VSI_Disk initialization parameter.

VSSet	VSIDisk <i>userid vdev</i> [FOrce REPlace]
-------	--

FOrce

Allows the definition to be re-established after having been , somehow, reset to Hex 0's. This is an error condition. The options should be used only under VSSI direction.

REPlace

Must be used to replace an existing definition.

Responses:

VSSI parm disk set to *vdev* owned by userid *userid*
Specify REPlace to overwrite existing definition.

VSSI parm disk was *vdev* owned by userid *userid*
VSSI parm disk set to *vdev* owned by *userid*

Chapter 3. Introduction and Concepts

VPARS is a software enhancement to VM that allows TPF to run in a virtual machine without modifying records on the TPF database. Records that are modified by TPF are maintained on a VPARS database consisting of one or more minidisks. "TPF" refers to the IBM Airline Control Program (ACP) or Transaction Processing Facility (TPF) operating system.

VPARS allows several virtual machines to share a common set of base TPF disks. This is called the "base TPF system" or the "TPF base" throughout this manual.

Each virtual machine that will share the base system has its own formatted VPARS database. In order to use VPARS, you need read-only links to the disks in the base TPF system. Other virtual machines may or may not be using the same TPF disks. Additional devices that you require for a test, such as tape drives or communication lines, can be attached by the VM operator or defined in the VM directory.

In a normal TPF environment, updated TPF records are written back to their original locations on the base TPF system. With VPARS, the write request is intercepted and the updated record is written to one of the minidisks in the user's VPARS database. The original record on the TPF base remains unchanged. A different virtual machine can retrieve the original record from the TPF base, update it, and write it to its own VPARS database.

All TPF records written to the system, including PILOT, OLD, and General File loads, are placed on the primary VPARS database. When a read is issued for a program or data record, VPARS searches its database directory. If the requested record is found in the directory, it is returned from the VPARS database; if not, it is retrieved from the TPF base disk. If the record is then modified and filed, it is written to the VPARS database. A subsequent request for the same record will result in the record being returned from the VPARS database. Multi-level databases allow several VPARS databases to be searched for a TPF record.

TPF programs are not aware that VPARS is intercepting and handling its I/O requests. Complete integrity of the shared TPF test system is maintained because the records on the shared TPF base are never altered.

VPARS reduces the amount of hardware required to run multiple TPF test virtual machines. Dedicated TPF disks are not required to run tests with different test requirements, such as Control Program, CCP, and Applications testing. Loosely coupled testing and multi-level VPARS databases are supported.

The number of virtual machines that can share a common TPF system using VPARS is theoretically unlimited. In practice, the number is restricted by the availability of other hardware that cannot be shared, and by performance considerations.

Backups and Restores

The various backup, restore, checkpoint, and clear functions of VPARS provide you with the ability to resume testing from any point, or to start with a clean VPARS database without affecting the shared TPF base.

At any time during a testing session, you can stop the virtual machine and back up the modified TPF records to tape. After the data has been backed up, you can continue testing, or use your virtual machine for a different test. Checkpoints can be set, and you can clear the records from the VPARS database at any time.

Chapter 4. Using VPARS

To use VPARS, you must have read/only links to the disks to be intercepted and read/write links to a formatted VPARS database. The formatted database consists of one or more minidisks of any size that have been formatted with the VPFMT command. These minidisks can be defined in your VM directory entry or linked dynamically. If you are using a read/only VPARS database, the links to your database disks can be read links.

TPF Database Processing

The TPF base system disks that are to have their I/O intercepted by VPARS must be linked read/only. If they are not defined in your VM directory entry, you can link them with the CP LINK command. If a TPF disk is linked *read/write*, VPARS will not intercept the I/O to that disk, and the data on the disk may be permanently modified. You should only get write links to the TPF disks if you want to write on them. VPARS will intercept I/O to TPF disks that are attached read/only, unless the userid is a V=R or V=F guest.

When you open your VPARS database, all *read/only* TPF disks are internally set to *read/write* (to allow VM's CCW translation to take place), and a flag is set for each disk to notify VPARS to intercept I/O to those devices. All intercepted I/O is processed by VPARS. Records written by TPF to an intercepted TPF disk are actually written to your VPARS database. The response from the QUERY VIRTUAL DASD (or Q V DASD) command shows which devices are being intercepted by VPARS.

If you link to additional TPF disks after you have opened your VPARS database, you can use the VPSET INTERCEPT ON command to notify VPARS. You can also set TPF disks back to read/only and turn the VPARS intercept flags off with the VPSET INTERCEPT OFF command. When you close your database, TPF disks that are being intercepted are reset to read/only.

VPARS Database Configurations

VPARS uses configuration files, which may be the same for all users, or different for each user. This allows your installation to choose the disk device numbers and other information used to open each database. These files can be changed and reloaded without stopping VM or VPARS.

A *configuration* consists of several pieces of database-related information, such as the base disk device number and the number of disks in the database. You can open a database with any configuration, as long as you have links to the proper VPARS database disks and the TPF base system disks. Your systems programming staff, or similar group, will define the VPARS configurations. You need to know what configurations are available to you, and which configuration(s) you should use for different kinds of TPF testing. The VPQUERY command will list the names and definitions of these configurations.

Multi-Level Databases

Multi-level (or "concatenated") VPARS databases allow several databases to be searched for a TPF record. The primary database is normally a read-write database. The rest are always read-only databases. Several users can share each read-only database. This may reduce the number of records required on each VPARS user's read/write database, since a shortload or longload (or any other data) can be written to a database which is then shared read-only among several VPARS users.

All users must link to the databases at the various levels using the same sets of minidisk addresses. In other words, if one user has a database open using a particular base disk address, any other users wanting to share the same database level must have the minidisks in the database linked at the same addresses.

The primary or first-level database can also be a read-only database, but your TPF system or application will not be able to write any records.

Loosely-Coupled Testing

VPARS supports testing of loosely-coupled TPF systems. All TPF virtual machines participating in a loosely coupled test write to the same VPARS database. Each virtual machine must have write links to all disks in the database. VPARS provides only the multi-write capability; record locking is the responsibility of the application. IBM's Limited Lock Facility PRPQ can be used to provide record locking. Under VM/ESA release 2 and above, the Multi-Path Lock Facility (MPLF) can provide record locking.

Testing Without a TPF Base System

If a VPARS configuration is defined as a NOBASE system, it is possible to test TPF applications without being linked to a base TPF system. The NOBASE feature can provide a TPF online environment that requires less disk space than a full restore of an online test system. This is accomplished by restoring only active online records; short term and unused long term pool record are not restored. However, setting up a NOBASE system can be complex. See "Testing Without a TPF Base System" on page 126 for more information on defining a NOBASE system.

PTV Operation

Description

The PTV mode of VPARS operation maintains a special directory for keypoint and PTV program records. Any records written with a record identifier of "CK" (keypoint) or "CV" (APTV program) and any program record with "PTV" as a program identifier are recorded in the special directory.

PTV mode can be used to eliminate the requirement for a Data Base Restore (DBR) tape during PTV processing. It can also be used to maintain keypoint records across database clears.

When PTV mode is active, all keypoint and PTV records are maintained in the special record directory on the VPARS database. These TPF records are not cleared from the database with the normal VPARS clear options. This allows keypoint information and PTV programs to be maintained across database clears. When PTV mode is set off, the keypoint records and the database are cleared.

Method of Operation

The modification to PTV processing is designed to allow PTV to run with or without a DBR tape. Only if PTV is running in a virtual machine under VPARS control and PTV mode is set on will the DBR tape be eliminated.

To use PTV mode for elimination of the DBR tape:

1. A file resident TPF program must have been allocated and loaded to your TPF system.
2. Three PTV programs must have been modified to enter the added program during PTV processing.

The added program and the modifications for the PTV programs are distributed on the VPARS distribution tape.

Testing IPL Text

All IPL records that are written to a TPF disk are saved on the VPARS database. You can use the VPIPLWR command or a TPF general file load to write the IPL text records. Normally the IPL records on the base TPF system are used when TPF is IPLed. To conserve space on the VPARS database, you can build your general file load disks without the IPL option.

Deleting Records

There are several ways to delete TPF records from a VPARS database. VPCLEAR and VPCLEAR CHECKPOINT will clear records from the database and recover the space used by the deleted records.

There are two ways to delete individual TPF records from a VPARS database. Records can be deleted using the VPUTIL CMS command when the VPARS database is closed, or by issuing a special channel program when the VPARS database is open. Using either method does not return the space held by the records released. Only the entry is removed from the VPARS record directory.

When the special CCW chain is used to delete records, it can be issued through Diagnose code x'20', Diagnose code x'A8', Start I/O (SIO), or Start Subchannel (SSCH). The I/O must be issued against the TPF disk that the record would be on if it was not on the VPARS database.

The CCW chain must contain a Seek x'07', Search x'31', TIC x'08' and Write x'05' CCWs. The CCHHR for the Seek and Search must be the CCHHR of the record to be deleted. Define Extent x'63' and Locate Record x'47' can also be used to select the record to be deleted. The write CCW must have a data length of six (6), the CCW Skip flag must be on, and the CCW address must point at the word 'DELETE'.

If VPARS finds the requested record on the database, its key is removed from the record directory, and channel end/device end is posted for I/O completion.

If the requested record is not found on the VPARS database, channel end, device end, and incorrect length are posted for I/O completion.

Chapter 5. VPARS Command Summaries

The following VPARS commands are described in this manual.

CP Commands

VPADD

The VPADD command adds a minidisk to your VPARS database. This disk can be a permanently assigned minidisk, a preformatted VPARS pool minidisk, or it can be a T-disk that you have formatted with VPFMT.

VPCLEAR

The VPCLEAR command clears some or all TPF records from a VPARS database. All records can be cleared, or the database can be cleared back to any checkpoint.

VPCLOSE

The VPCLOSE command closes the VPARS database. The data base must be closed before it can be dumped to tape, restored from tape, or formatted. When you log off of VM, VPARS automatically closes your database.

VPFCLOSE

The VPFCLOSE command forces another user's VPARS database to be closed. This is a privileged command, and is restricted to class B users (unless your installation has changed it to another command class).

VPINIT

The VPINIT command initializes or reinitialize the VPARS system defaults. You may want to reinitialize the VPARS defaults after you have modified the VPSYSTEM DEFAULTS file.

VPIPL

The VPIPL command performs a CP IPL command. Also, VPIPL has additional parameters, and can clear a database before an IPL. This command passes control to normal CP IPL processing after processing any VPARS options.

VPLINK

The VPLINK command links one or more minidisks from a pool of preformatted VPARS minidisks. If your database is already open, use VPADD to add the newly-linked pool disk(s) to your database.

VPOPEN

The VPOPEN command opens your VPARS database and flags your TPF database disks to be intercepted by VPARS. You normally open your VPARS database just before you IPL TPF.

VPQUERY

The VPQUERY command displays the status of your VPARS database, your I/O activity, the definitions of VPARS configurations, etc. See the description of VPQUERY for details.

VPREMOVE

The VPREMOVE command removes inactive VPARS minidisks from your database.

VPSET

The VPSET command alters any of several VPARS options. See the description of VPSET for details.

VPTEST

The VPTEST command runs a VPARS configuration file through syntax checking to verify the validity of the statements.

CMS Commands

VPBKUP

VPBKUP writes a physical backup of a VPARS database to tape. The backup can be restored to any VPARS database with the same number and size of minidisks.

VPBXBMAP

VPBXBMAP reads TPF pool bit map records from tape and creates a CMS file. This file is used with the pool section table, created by VPBXPLTB, to identify active pool records when building a NOBASE VPARS database with VPBXREST.

VPBXPLTB

VPBXPLTB is assembled against TPF macros to build a pool section table. The pool section table identifies the starting and ending track numbers of pool areas on a TPF system disk. It is used with the pool bit map directories, created by VPBXBMAP, to identify active pool records when building a NOBASE VPARS database with VPBXREST.

VPBXREST

VPBXREST restores records from a TPF capture tape. It was written to build NOBASE VPARS databases.

VPFMT

VPFMT formats a minidisk for use as a VPARS database disk.

VPLOAD

VPLOAD loads records from a tape created by VPUNLD. If the VPARS database is open, the records will be placed on the VPARS database. If the database is closed and you have write links to the base TPF system, the records will be written to the base system disks.

VPIPLWR

VPIPLWR will write IPL bootstrap and text records to track zero of a specified disk. IPL text can be written in DSF (O/S) format or TPF format. If the VPARS database is open, the IPL records will be written to the VPARS database.

VPREST

VPREST restores a backup of a VPARS database.

VPUNLD

VPUNLD unloads TPF records from a VPARS database. Records written to tape by VPUNLD can be restored to a TPF base system or to an open VPARS database.

VPUTIL

VPUTIL performs utility functions, such as listing, printing, or deleting TPF records on a VPARS database. VPUTIL can also move records from a VPARS database to a TPF base or another VPARS database.

Chapter 6. VPARS CP Commands

This section describes the VPARS CP commands. The format, use, and the normal responses of each command are explained. Error responses, such as disks not being attached, invalid keywords, etc., are not listed here. You can find explanations of error messages in the "Messages" section of this manual, or by using the CMS Help facility.

VPACC

Use the VPACC command to have CP access the private VPARS mini-disk associated with your userid or privilege classes.

VPADD	<i>fmode</i>
-------	--------------

fmode

The file mode to access the minidisk at.

Usage Notes:

1. What minidisk to access is determined by the PARM_USER and/or PARM_CLASS keywords in the VPSYSTEM DEFAULTS file.
2. The filemode requested must be available. VPACC will not release the minidisk currently accessed as *fmode*

VPADD

Use the VPADD command to add a VPARS formatted minidisk to an open database. This will increase the capacity of the database.

VPADD	[<i>n</i>] [ALL]
-------	-------------------------

n the number of minidisks to be added. If you do not specify a number or the word 'ALL', the default is to add 1 minidisk.

ALL adds all minidisks that are in the VPARS database device number range to the databases.

Usage Notes:

1. If more than one user has a database open (loosely-coupled), all users must have link to a minidisk before VPADD will add it to the database.
2. Pool disks can only be used in single-user databases. If you use VPLINK to link a pool disk, you cannot add this disk to a multi-user (loosely coupled) database. If you need to add capacity to a loosely-coupled database, you must use the CP LINK command to link to a (non-pool) minidisk. All users who are coupled to the database must link this disk read/write before VPADD will add it to the database.
3. For a single-user database, you can use VPADD to add space to your VPARS database if you receive messages that your database is approaching capacity. This can be done while your TPF test is running. You can link to one or more preformatted VPARS database minidisks using the VPLINK or CP LINK commands, and use VPADD to add them to your database.
4. If your VPARS database becomes 100% full, all write I/O that is intercepted by VPARS will be rejected with a command reject. To continue testing, you must either increase the capacity of your database by adding additional database minidisks, clearing your database, or clearing to a previously-set checkpoint.

Response:

```
vdev was added to your VPARS database  
...
```

(followed by VPARS status).

VPCLEAR

Use the VPCLEAR command to clear TPF records from your VPARS database.

VPCLEAR	[ALL] [CHeckpoint [<i>n</i>]] [CHkpoint [<i>n</i>]] [CLearcheck [<i>n</i>]] [NOReset] [RESTRicted <i>cfgname</i>] [CLearchk [<i>n</i>]] [DAtabase]
---------	---

ALL clears all TPF records from the database, and turns off checkpointing and PTV mode.

CHeckpoint or CHkpoint

clears all TPF records added to your database after the last checkpoint was set. If a number *n* is specified, the database is cleared to that checkpoint. If a checkpoint clear is specified and no checkpoints are set, a database clear is performed.

CLearcheck or CLearchk

performs a checkpoint clear, removes all checkpoints, and turns off checkpointing.

DAtabase

clears all TPF records and checkpoints from your database. PTV records are not cleared, and PTV mode is not turned off.

NOReset

allows your virtual machine to continue running after the clear completes. If you don't specify NORESET, your virtual machine is reset and you must re-IPL to continue testing. You should only use NORESET while running CMS in your virtual machine.

RESTRicted *cfgname*

is required if the clear function is restricted. It is not allowed if the clear function is not restricted. The configuration name entered must match the configuration name for the database being cleared.

Usage Notes:

1. If no options are given, a VPCLEAR CHECKPOINT is performed. If no checkpoints are set, VPCLEAR CHECKPOINT performs a VPCLEAR DATABASE.

2. NORESET is provided to allow CMS to continue running after a VPCLEAR. If you do not specify NORESET, your virtual machine will be reset and you must re-IPL to continue. You should only use NORESET while running CMS in your virtual machine.

Responses:

```
VPARS database clear complete  
or  
VPARS checkpoint clear complete
```

VPCLOSE

Use the VPCLOSE command to close your VPARS database and turn off interception of your TPF base system I/O.

VPCLOse	[NOReset]
---------	-------------

Noreset

causes VPARS to close the database without resetting the virtual machine. You should only use NORESET while running CMS in your virtual machine.

Usage Notes:

1. You must close the database to use many of the VPARS CMS utilities, such as VPBKUP, VPUNLD, VPREST, etc.
2. If you IPL TPF after closing the database, TPF will attempt to write to the base system. If the base disks are not linked read/write, TPF will receive an error on every write attempt.
3. NORESET is provided to allow CMS to continue running after a VPCLOSE. If you do not specify NORESET, your virtual machine will be reset and you must re-IPL to continue testing. You should only use NORESET while running CMS in your virtual machine.

Response:

VPARS database is closed

VPFCLOSE

Use the VPFCLOSE command to force another user's database closed. This is normally a class B command.

VPFCLOSE	{ Key <i>dbkey</i> } { Userid <i>userid</i> }
----------	--

Key *dbkey*

closes the databases of all users who have the database with the given key open. The key is an eight-character hexadecimal number used to uniquely identify each open database in the system. It is based on the real location of the first database minidisk.

Userid *userid*

closes the given user's database.

Responses:

VPARS close scheduled for *userid*

This message is received for each user whose database is being closed.

RVPSET160W VPARS force close scheduled by *userid*

This message is received by all user(s) whose databases are being closed. In this message, *userid* is the ID of the user issuing the VPFCLOSE command.

VPINIT

Use the VPINIT command to initialize the online system-wide VPARS defaults from the VPSYSTEM DEFAULTS configuration file. This is normally a class B command.

VPINIT	
--------	--

Usage Notes:

1. The VPINIT command places the information from the VPSYSTEM DEFAULTS configuration file online. VPINIT is normally used after you modified the VPSYSTEM DEFAULTS file on the VSSI parm disk.
2. VPINIT is not required each time you IPL VM. Data from the VPSYSTEM DEFAULTS configuration file is automatically placed online the first time a user issues a VPARS command after a VM IPL.
3. The pool disk userid, default pool disk size, default VPARS configuration name, and most other items from VPSYSTEM DEFAULTS are activated. However, the maximum number of VPARS users allowed to open a database (MAXUSERS) is not changed if the system has already been initialized. This allows you to restrict VPARS use by setting MAXUSERS to 0 or some other small number, and it won't be changed during a VPINIT. The class B VPSET MAXUSERS command can be used to change the MAXUSERS limit at any time.

Response:

```
VPARS system initialization complete
```

VPIPL

Use the VPIPL command to specify VPARS options during an IPL.

VPIPL	[VCheckclr [<i>n</i>]] [VChkclear [<i>n</i>]] [RESTRicted <i>cfgname</i>] UNit <i>vdev</i> [<i>options</i>] [VCLear]
-------	--

RESTRicted *cfgname*

is required if the clear function is restricted and any type of clear is requested. It is not allowed if the clear function is not restricted. The configuration name entered must match the configuration name for the database to be cleared.

VCheckclr or **VChkclear**

causes TPF records added or updated after the last checkpoint to be cleared before the IPL. If a checkpoint number *n* is entered, the database is cleared to that checkpoint.

VCLear

causes all TPF records on the VPARS database to be cleared before the IPL.

UNit *vdev*

the virtual device number or system to be IPLed. The UNit keyword must be specified, and must be the last one before the IPL options.

options

any normal IPL parameters to be passed to the CP IPL command.

Usage Notes:

1. Any standard options of the CP IPL command can be specified.

Responses:

Normal responses for an IPL will be displayed.

VPLINK

Use the VPLINK command to request that a minidisk be linked from the VPARS pool.

VPLInk	<i>devicetype</i> [<i>n</i>] [Cfg <i>cfgname</i>] [CONFig <i>cfgname</i>] [Size <i>nn</i>]
--------	---

devicetype

the device type (3380, 3390, etc.) to be linked from the pool of formatted VPARS database minidisks.

n the number of disks to link.

CONFig *cfgname* (or) **CFg** *cfgname*

gives a configuration name from which to get the default pool minidisk size.

Size *nn*

specifies the size of the requested minidisk in cylinders. If you do not specify a size, VPARS will assign a minidisk of the size defined in your default configuration. If you request a size of zero, the first available minidisk of any size is assigned.

Usage Notes:

1. Your default VPARS configuration specifies a default size for pool minidisks. VPLINK searches the pool directory for minidisks of the requested disk type and size. If one is found that is not in use, it will be linked at the next available device number in the database minidisk range.
2. If no disks of the default size are available, you may want to reissue the command specifying SIZE 0. This will cause VPLINK to search for a pool disk of any size.
3. If your database is already open, you can add the newly-linked disk to the database with the VPADD command.
4. You should not use VPLINK to link minidisks to be added to a loosely-coupled database (when more than one user has the same database open in COUPLED mode). VPADD will not add a minidisk to a loosely-coupled database. If you need to add capacity to a loosely-coupled database, you should use the CP LINK command to link to a (non-pool) minidisk. All users who are coupled to the database must link to the disk read/write. You can then use VPADD to add the minidisk to the database.

5. If you close your VPARS database after TPF records have been written to a pool disk, you must have a link to the same pool disk in order to re-open your database. However, when you sign off or detach the pool disk, it will become available for other users to link, and it may get overwritten by another user. If you need to preserve the contents of the database after adding a pool disk, you should use VPBKUP or VPUNLD to back up the data to tape. You can then restore the database before beginning your next testing session.

Response:

VPARS pool disk *vdev* on *volser* was linked as *vdev*

VPOPEN

Use the VPOPEN command to open your VPARS database.

VPOPen	[CCT <i>concatname</i>] [CFG <i>configname</i>] [CONCat <i>concatname</i>] [CONFig <i>configname</i>] [CHeckpoint [<i>n</i>]] [CHkpoint [<i>n</i>]] [CLear] [CLRdb] [COUpled] [DEVtable <i>devname</i>] [DEVtbl <i>devname</i>] [NOCONCat] [NOCONFig] [NOCCt] [NOCFg] [NOReset] [RDOOnly] [RESTRicted <i>cfname</i>]
--------	---

The VPOPEN command has several possible options. You should be familiar with VPARS database configurations, multi-level databases, and the names of the database configurations that are available at your installation.

CONCat *concatname* (or) **CCT** *concatname*
opens the specified multi-level concatenated database. A concatenation consists of a series of configuration names.

CONFig *configname* (or) **CFg** *configname*
opens the database with the specified configuration. A VPARS database configuration specifies several options, including the starting minidisk address and the number of minidisks in the database.

CHeckpoint *n* (or) **CHkpoint** *n*
opens the database with the TPF records that were present when checkpoint *n* was set.

CLear
performs a VPCLEAR ALL and opens the database.

CLRdb
performs a VPCLEAR DATABASE and opens the database.

COUpled

opens the database, and allows other TPF virtual machines to open the same VPARS database in read/write mode. This option is provided to support TPF loosely coupled testing. If one or more users have already opened this database with the COUPLED option, you will "join" the already open database as an additional user. All users with this database open will be able to write to it. If another user opened the database for output, but did not specify COUPLED during the open, you will not be able to couple to it. See the usage notes for more information.

DEVtbl *devname*

opens your database using the specified TPF device table, and the TPF format tables associated with it.

NOCONCat (or) **NOCCt**

allows you to open a database without using the default concatenation, if one is defined in your system.

NOCONFig (or) **NOCFg**

allows you to open a concatenated database without a read/write database. This option is only valid if there is a default concatenation in your system, or if you specify a concatenation name with the CONCAT (or CCT) option.

NOReset

does not reset your virtual machine when the CLEAR option is requested. You should only use NORESET while running CMS in your virtual machine.

RDOnly

opens your primary database read/only. Any write I/Os intercepted by VPARS will be rejected (with a "command reject"). Your VPARS database disks can be linked read/only if you specify this option.

RESTRicted *cfgname*

is required if the clear function is restricted and any type of clear or checkpoint open is requested. It is not allowed if the clear function is not restricted. The configuration name entered must match the configuration name for the database to be cleared.

Usage Notes:

1. If you do not specify a configuration name, the VPARS user default table (from VPSYSTEM DEFAULTS) is searched for a userid prefix which matches your userid. If one is found, that configuration will be used. This capability allows different groups of VPARS users to have different VPARS defaults.
2. If your userid prefix is not found in the system defaults table, the system-wide default configuration name will be used. If there is no system-wide default configuration name, you must specify a configuration name in order to open a VPARS database.

3. You must have your VPARS database disks linked at the device numbers (addresses) specified in the configuration definition.
4. If you specify a concatenation name, or if there is a default concatenation name, the databases in the concatenation determine your read/only database levels. Your default configuration name, or a configuration name you specify, determines your read/write database. If you do not want your primary read/write database to be included, you must specify NOCONFIG along with CONCAT. If you want your primary database to be included, but opened read/only, specify RONLY. In either of these cases, you will have a multi-level read/only database.
5. The TPFDEV table name in your default (or specified) configuration determines which TPF disks will be intercepted, and which TPF format tables will be used for record length checking.
6. When you open your database, all minidisks in the database device range for the selected configurations are made part of the database.
7. If you open your VPARS data base read/write, read/only TPF disks, at addresses defined in your TPFDEV device table, are made read/write, this allows CCW translation to process write requests. A flag is set for each TPF disk to cause VPARS to intercept I/O requests. If you open your VPARS data base read/only your TPF disks are not made read/write and any write requests will be rejected by CCW translation.

The CP Query Virtual Dasd can be used to determine if a disk is being intercepted by VPARS. The word VPARS will be in the display for each disk that is being intercepted.

8. You should have read/only links to the TPF disks before using VPOPEN. If the disks are linked or attached read/write, VPOPEN and VPSET INTERCEPT ON will not flag them to be intercepted. If you link TPF disks read/write, be sure that you really want to write on them.
9. To open to a checkpoint, you must have links to all VPARS database minidisks that you had when the checkpoint was set.
10. The COUPLED parameter is supplied to support TPF loosely coupled testing. All TPF virtual machines participating in a loosely coupled test write to the same VPARS database. Each virtual machine must have write links to all disks in the VPARS database, and each user must specify the COUPLED option on the VPOPEN command. VPARS provides only the multi-write capability; record locking is the responsibility of the application. IBM's Limited Lock Facility PRPQ can be used to provide record locking.
11. When more than one user is coupled to a database, certain commands such as VPCLEAR are not allowed.
12. If one or more users have a database open read/only, and you want to open it read/write, the VPOPEN command will complete only if you are authorized. VPARS checks your command privilege class against the classes listed on the OPEN_RO_FOR_Output list in the configuration file used to open the

database. If you have the correct command class or if your installation has set the class to a general user class such as G, you will be allowed to open the database for output.

13. If you are opening a multi-level database with the read/write level loosely coupled with other users, you *may* have to have the same set of read/only database disks as the other users. If not, this mismatch could cause incorrect data records to be returned to your TPF application. Whether or not this is allowed is controlled by the LCP_unlike option in the configuration file used to open the database.

Response:

VPARS database is open

(followed by VPARS status)

VPQUERY

The VPQUERY command displays VPARS database activity, TPF I/O activity, configuration and concatenation definitions, and other information. Each subcommand is described separately on the following pages with any additional options.

VPQuery	[ACTivity]	[select A]	
	[BUffers]	[select A]	
	[CCt]	[select E]	
	[CFg]	[select D]	
	[CONCat]	[select E]	
	[CONFig]	[select D]	
	[DEFaults]		
	[DEVtbl]	[select E]	
	[FMTtbl]	[select E]	
	[KEY]	[select F]	
	[LOKqueue]	[select A]	
	[MAXusers]		
	[MDiskpool]		
	[OPen]	[select C]	
	[SStatus]	[select B]	
	[TPFstatus]	[select A]	
	[Users]	[select A]	
	[VPstatus]	[select A]	
	select A:	select B:	select C:
	ALL	CFg <i>cfgname</i>	NOMSG
	CFg <i>cfgname</i>	CONFig <i>cfgname</i>	Userid <i>userid</i>
	CONFig <i>cfgname</i>	KEY <i>dbkey</i>	
	KEY <i>dbkey</i>	LEvel <i>n</i>	
	LEvel <i>n</i>	Userid <i>userid</i>	
	Userid <i>userid</i>		
	select D:	select E:	select F:
	ALL	NAMES	CFg [<i>cfgname</i>]
	NAMES	<i>concatname</i>	CONFig [<i>cfgname</i>]
		<i>cfgname</i>	VDev <i>vdev</i>
		<i>devname</i>	
		<i>fmtname</i>	

VPQUERY ACTIVITY

Use the VPQUERY ACTIVITY command to display TPF and VPARS database activity for the previous second.

VPQuery	Activity	[ALL] [CFg <i>cfgname</i>] [CONFIg <i>cfgname</i>][FULL] [KEy <i>dbkey</i>] [INterval] [LEvel <i>n</i>] [Userid <i>userid</i>]
---------	----------	---

ALL displays the activity for all open VPARS databases for all users.

CONFIg *cfgname* (or) CFg *cfgname*

displays the activity for the database that was opened with the requested configuration name. If you do not also specify a userid, your open databases will be searched for the named configuration.

FULL displays the activity since the VPARS database was opened or cleared.

INterval

displays the activity for the interval time specified in the configuration used for open or with the VPSET command.

KEy *dbkey*

displays the activity for the database with the requested key. The key is an eight-character hexadecimal number used to uniquely identify each open database in the system. It is based on the real location of the first database minidisk.

LEvel *n*

displays activity for the database at the requested level in a multi-level database. This can be one of your open databases, or another user's database if you also specify a userid.

Userid *userid*

displays the activity for all open databases for the requested user. If you also specify a level number or a configuration name, you will see activity for one database.

Usage Notes:

1. The default is to display the activity for all of your open databases.
2. Key is mutually exclusive with any other option.

3. Level and Config are mutually exclusive.

Response:

DB key	Users	TPFreq	Added	Found	Updated	VPio	VPread	VPwrite
10470200	2	9	2	1	1	0	0	0

DB key

the VPARS database key. The key is an eight-character hexadecimal number used to uniquely identify each open database in the system. It is based on the real location of the first database minidisk.

Users

the number of users who currently have this database open.

TPFreq

the number of TPF I/O requests in the requested interval.

Added

the number of TPF records added to the VPARS database.

Found

the number of requested TPF records the were found on the VPARS database.

Updated

the number of TPF records that were already present on the VPARS database that were updated (rewritten).

VPio

the number of I/O requests to the VPARS database.

VPread

the number of read requests to the VPARS database.

VPwrite

the number of write requests to the VPARS database.

VPQUERY BUFFERS

Use the VPQUERY BUFFERS command to display the system buffer usage for some or all VPARS databases. The information displayed by this command is intended for use in tuning the VPARS system, and may not be meaningful to the general VPARS user.

VPQuery	Buffers	[ALL] [CFg <i>cfgname</i>] [CONFig <i>cfgname</i>] [KEy <i>dbkey</i>] [LEvel <i>n</i>] [Userid <i>userid</i>]
---------	---------	--

ALL displays buffer usage for all open VPARS databases.

CONFig *cfgname* (or) CFg *cfgname*

displays buffer usage for the database that was opened with the requested configuration name. If you do not also specify a userid, your open databases will be searched for the named configuration.

KEy *dbkey*

displays buffer usage for the single database with the requested key. The key is an eight-character hexadecimal number used to uniquely identify each open database in the system. It is based on the real location of the first database minidisk.

LEvel *n*

displays buffer usage for the database at the requested level in a multi-level database. This can be one of your open databases, or another user's database if you also specify a userid.

Userid *userid*

displays buffer usage for all open databases for the requested user. If you also specify a level number or a configuration name, you will see activity for one database.

Usage Notes:

1. The default is to display the buffer usage for all of your open databases.
2. Key is mutually exclusive with any other option.
3. Level and Config are mutually exclusive.

Usage Notes:

1. VPARS buffers are 4096-byte pages that are allocated from system virtual storage. They contain record directories and directory indexes that support the VPARS database structure, along with records that are being read in or written out to disk, for all VPARS users.
2. The total number of buffers listed under BUFS includes some buffers that are not displayed elsewhere in the result. These include one buffer for the "high index", and buffers for the current large and small output blocks.

Response:

DB key	Users	BuFs	Max	XMax	IIs	DIIs	RDs	DBs	Locked	ActIO
10450200	1	212	2005	0	5	23	87	94	4	1
10450210	2	34	1000	10	1	10	10	10	0	0

System buffers in use by selected databases: 246

DB key

the VPARS database key. The key is an eight-character hexadecimal number used to uniquely identify each open database in the system. It is based on the real location of the first database minidisk.

Users

the number of users who currently have this database open.

BuFs the total number of buffers currently in use for this database.

Max the largest number of buffers that VPARS will allocate for this database.

Xmax

the number of times the total number of buffers in use for the database exceeded the Max value. This happens when buffers need to be allocated in response to TPF I/O requests, and the database is already using its maximum number of buffers. When a database exceeds maximum buffers, new TPF I/O requests against that database are suspended until buffers are released at the completion of the currently outstanding VPARS I/O.

IIs the number of buffers currently in use for "index indexes".

DIIs The number of buffers currently in use for "directory indexes".

RDs the number of buffers currently in use for "record directories".

DBs the number of buffers currently in use for "data blocks".

Locked

the number of locked buffers. Buffers are locked for active I/O, and while they are being updated.

ActIO

the number of buffers queued for VPARS I/O.

VPQUERY CONCAT

Use the VPQUERY CONCAT command to display the definition of a multi-level database concatenation, or the names of all the concatenations the user issuing the command has access to.

VPQUERY	CONCat	[NAMEs]
	CCt	[<i>concatname</i>]

concatname

displays the definition of a specific concatenation.

NAMEs

displays the names of all concatenations the user can access.

Usage Notes:

1. For the *concatname* option, VPARS will display the names of the configurations that make up the concatenation, along with their VPARS database minidisk addresses. This identifies which minidisks you need to link in order to use VPOPEN with this concatenation. You can display more detailed information for any of these configurations with VPQUERY CONFIG.
2. The databases in the concatenation are displayed in the order they are searched for records, that is, top-down.
3. If any configuration named in the concatenation cannot be found, you will see "Not found" listed under VP Database for that configuration.

Response:

VP Config	VP Database	VP Config	VP Database	VP Config	VP Database
CONFIGA	0FA0-0FAF	CONFIGB	0FB0-0FBF	CONFIGC	0FC0-0FCF
CONFIGD	0FD0-0FDF				

VP Config

the names of the configurations in the concatenation.

VP Database

the device number ranges of the disks that will be used for the database.

VPQUERY CONFIG

Use the VPQUERY CONFIG command to display the definition of a VPARS database configuration, or the names of all the configurations the user issuing the command has access to.

VPQUERY	CONFig CFg	[<i>cfgname</i>] [ALL] [NAMEs]
---------	---------------	--

cfgname

displays the definition of a specific configuration.

ALL displays the names and definitions of all configurations defined in the system.

NAMEs

displays the names of all configurations the user has access to.

Usage Notes:

1. If no config name is specified, VPQUERY CONFIG will display your default VPARS configuration. If a default is not defined for your userid, it will display the system-wide default configuration. In other words, you will see the configuration that will be used by VPOPEN if you do not specify a configuration name.

Response:

```
VP Config  VP Database  Dev Table  Autostat  Pmd sz  Min Q  Max Q  Base  Clear
CONFIGA   0FA0-0FAF   TPF1DEV   100       0       10    1000  Yes   Norm
```

VP Config

the name of the configuration.

VP Database

the device number range of the disks that will be used for the database.

TPF Table

the name of the TPF device table that is used when the database is opened. This table provides the device number ranges of the disks whose I/O will be intercepted when the database is open.

Autostat

the time interval for automatic VPARS status, in minutes.

Pmd sz

the default pool minidisk size used by the VPLINK command.

Min Q

this is a tuning parameter, and represents the minimum number of buffers on record directory and data block queues.

Max Q

this is a tuning parameter, and represents the maximum number of buffers on record directory and data block queues.

Base

whether or not a TPF base system is associated with this configuration. "No" indicates that this is a "NOBASE" VPARS configuration. See "Testing Without a TPF Base System" on page 126 for more information.

Clear

whether or not the database clear function is restricted. "Norm" means the clear function is normal, or not restricted; "Rest" means the clear function is restricted.

VPQUERY DEFAULTS

Use the VPQUERY DEFAULTS command to display the system-wide VPARS default configuration information.

VPQUERY	DEFaults
---------	----------

Usage Notes:

1. VPQUERY DEFAULTS takes no parameters. It displays the system-wide default configuration information that is contained in VPSYSTEM DEFAULTS. It will show the default configuration name that is associated with each userid prefix. A userid prefix of VPTPF, for example, "matches" any userid that begins with "VPTPF". A userid prefix can be up to eight characters in length.
2. The response shows the name of the system default configuration, which is selected if the user's ID does not match any of the defined prefixes. It also shows which configuration names will be selected by default for a VPOPEN command issued by users whose IDs begin with the listed prefixes. A user can override the defaults with options on the VPOPEN command.

Response:

Config	Concat	Format	Max users		
CONFIGA	CONCAT1	TPFDEV1	150		
Prefix	Config	Concat	Prefix	Config	Concat
VP	DEFCFG	DEFCCT	V24	CFG24	CCT24
V31	CFG31	(none)	V32	SPECIAL	CCT32

Config

the name of the system default configuration.

Concat

the name of the system default concatenation.

Format

the name of the system default format table.

Max users

the default number of users allowed to open VPARS databases in the system. This number is the default at system startup; it may have been changed by VPSET MAXUSERS. VPQUERY MAXUSERS will show you the current maximum user limit.

Prefix Config Concat

The remaining sets of names are userid prefixes and the default configuration and concatenation name to be used for a user with a userid that starts with the given userid prefix.

VPQUERY DEVTBL

Use the VPQUERY DEVTBL command to display the definition of a TPF device table, or the names of all device tables the user issuing the command has access to.

VPQUERY	DEVtbl	{ devname } { NAMEs }
---------	--------	--------------------------

devname

displays the definition of a specific device table.

NAMEs

displays the names of all TPF device tables the user has access to.

Usage Notes:

1. The TPF device table that is used when you open a database determines which disks will have I/O intercepted by VPARS. Any read/only disk that appears in any range in the device table will have its I/O intercepted.
2. The format table associated with each range may or may not provide record length checking for disks in that range. Different ranges can use the same format table. See VPQUERY FMTTBL for more information.

Response to query by name:

VDEV range	Fmt table	VDEV range	Fmt table	VDEV range	Fmt table
0300-031F	FT3380A	0400-040F	FT3380A	0800-085F	FT3380B
0900-095F	FT3380B				

VDEV range

a range of device numbers (addresses). All disks in any of these ranges should be part of the base TPF system.

Fmt table

the name of the TPF format table to be used for disks in this range.

VPQUERY FMTTBL

Use the VPQUERY FMTTBL command to display the definition of a TPF format table, or the names of all format tables the user issuing the command has access to.

VPQUERY	FMTtbl	{ <i>fmtname</i> } { NAMES }
---------	--------	---------------------------------

fmtname

displays the definition of a specific TPF format table.

NAMES

displays the names of all TPF format tables the user has access to.

Usage Notes:

1. The TPF device table that is used when you open a database determines which format table is used.
2. A format table may or may not provide record length checking for disks in that range. If record length checking is provided, and a TPF application writes a wrong length record, VPARS will reject the I/O with an "incorrect length" indication. This is what would happen in a native TPF system. If you do not provide record lengths in a format table, VPARS is not able to determine the correct record length for a record. In this situation, VPARS will not check the record length the first time a record is written.

Response for query by name:

Scyl track	Ecyl track	Rsize	Rincr	Rbits
0000 0001	0005 0000	1055	8	1
0005 0001	0040 0005	4096	1	0
0040 0006	0150 0001	381	4	0
0150 0002	0238 0013	1055	8	1
0238 0014	0245 0008	4096	1	0

(etc.)

Scyl track

the starting cylinder and track, in decimal, for a range of TPF records.

Ecyl track

the ending cylinder and track, in decimal, for a range of TPF records.

Rsize

the size of the TPF records in this range.

Rincr

the record increment for TPF records in this range.

Rbits

the value of the low-order two bits of the record numbers for TPF records in this range.

VPQUERY KEY

Use VPQUERY KEY to find the VPARS key for a specific database, and whether or not that database is open by anyone.

VPQuery	KEY [CFg [<i>cfgname</i>]] [CONFig [<i>cfgname</i>]] [VDev <i>vdev</i>]
---------	--

CONFig [*cfgname*] (or) CFg [*cfgname*]

displays the database key and the open/closed status of the database with the requested configuration name. If the configuration name is omitted, your default name is used.

VDev *vdev*

displays the database key for the specified virtual device number. The device number should be the first minidisk in a database.

Usage Notes:

1. If you do not specify an option, the key for your default configuration will be shown.
2. If you use the VDEV option, VPARS will build an eight digit key using the requested device number. The device must be linked. If you use the CONFIG or CFG option, VPARS will build a key from the minidisk you have linked at the first minidisk device number for that configuration.
3. The key is based on the real location of the first minidisk in the VPARS database. The first four digits are the device number (address) of the real disk containing the first minidisk. The last four digits are the starting cylinder number of the minidisk on the real disk. The key is then used to check if this database is open by any users.
4. If a virtual device number is requested that is not the first minidisk in a VPARS database, a key will still be derived, but it will never be found as part of an open database. This is true even if the device number supplied is another minidisk (not the first) in a database, since the key used by VPOPEN is based on the first minidisk in the database.

Response:

Database key *dbkey*, base disk *vdev*, is open for *inout* with *n* users
or
Database key *dbkey*, base disk *vdev*, is not open

dbkey

the database key. The key is an eight-character hexadecimal number used to uniquely identify each open database in the system. It is based on the real location of the first database minidisk.

vdev the virtual device number of the first minidisk in the database.

inout

"Input" or "Output"

n the number of users using the database.

VPQUERY LOCKQUEUE

Use the VPQUERY LOCKQUEUE command to display lock statistics for the Record Directory and Data Block queues.

VPQuery	LOckqueue	[ALL] [CFg <i>cfgname</i>] [CONFIg <i>cfgname</i>][FULL] [KEy <i>dbkey</i>] [INterval] [LEvel <i>n</i>] [Userid <i>userid</i>]
---------	-----------	---

ALL displays the lock queue statistics for all open VPARS databases for all users.

CONFIg *cfgname* (or) CFg *cfgname*

displays the lock queue statistics for the database that was opened with the requested configuration name. If you do not also specify a userid, your open databases will be searched for the named configuration.

FULL displays the lock queue statistics since the VPARS database was opened or cleared.

INterval

displays the lock queue statistics for the interval time specified in the configuration use for open or with the VPSET command.

KEy *dbkey*

displays the lock queue statistics for the database with the requested key. The key is an eight-character hexadecimal number used to uniquely identify each open database in the system. It is based on the real location of the first database minidisk.

LEvel *n*

displays lock queue statistics for the database at the requested level in a multi-level database. This can be one of your open databases, or another user's database if you also specify a userid.

Userid *userid*

displays the lock queue statistics for all open databases for the requested user. If you also specify a level number or a configuration name, you will see lock queue statistics for one database.

Usage Notes:

1. The default is to display the lock queue statistics for all of your open databases.
2. Key is mutually exclusive with any other option.
3. Level and Config are mutually exclusive.

Response:

DB Key	User(s)	Lock	Requests	Delayed	%dly	Avg queue:	Length	Time
08720340	VSSIIVP1	RDQ	1000	30	3%		0	0.00
08720340	VSSIIVP1	DBQ	700	3	1%		0	0.00

DB key

the VPARS database key. The key is an eight-character hexadecimal number used to uniquely identify each open database in the system. It is based on the real location of the first database minidisk.

Users

the number of users who currently have this database open.

Lock

RDQ for Record Directory Queue lock, DBQ for Data Block Queue lock.

Requests

the number lock requests that have been done in the requested interval.

Delayed

the number lock requests that were delayed.

%dly

the percentage of lock requests that were delayed.

Length

the average queue length for delayed requests.

Time

the average time a lock request was delayed.

VPQUERY MAXUSERS

Use the VPQUERY MAXUSERS command to display the maximum VPARS user limit.

VPQUERY	MAXusers
---------	----------

Usage Notes:

1. The maximum VPARS user limit is the systemwide number of users who are allowed to open a database. If this number is reached, VPARS will not allow any more users to open a database until the number of open databases falls below this limit. The class B VPSET MAXUSERS command will change the limit.

Response:

Maximum number of VPARS users set to *n*. Currently *n* active users.

VPQUERY MDISKPOOL

Use the VPQUERY MDISKPOOL command to display all VPARS pool minidisks in use in the system.

VPQUERY	MDiskpool
---------	-----------

Response:

```
VPARS pool disk vdev, on volser, is held by userid, as vdev  
VPARS pool disk vdev, on volser, is in use by the database with key dbkey  
  
(etc.)
```

The second form of the response is given if the user who added the pool minidisk to a database has detached it. The pool disk may still be in use by a database.

VPQUERY OPEN

Use the VPQUERY OPEN command to show whether a database is open or closed.

VPQUERY	OPen [Userid <i>userid</i>] [NOMsg]
---------	---

Userid *userid*

displays VPARS open/closed status for another user.

NOMsg

displays the open/closed status in the return code, without displaying a result.

Usage Notes:

1. If you don't specify an option, the result will show whether or not you have a database open.
2. The NOMSG option is intended to be used from an exec or CMS module. With NOMSG, a return code of 0 means that the database is not open. If you (or the user you specified) have an open database, you will get a nonzero return code. This consists of a fullword where the low-order halfword is the address of the first minidisk in the primary database, and the high-order halfword is the maximum number of minidisks in the database. In addition, the x'80' bit in the high-order byte is set to 1. The x'40' bit is set to 1 if it is an output database. The x'20' bit is set to 1 if PTV mode is on. If you use the NOMSG option from the CMS command line, CMS will convert the return code to decimal, and only display the last 5 digits.
3. A response similar to the one shown below is displayed if you don't specify NOMSG.

Response:

VP Config	VP Database	Status	Users	DB Key	Base	Echo	ReqOL	Clear
VPCFG120	1200-120F	Output	1	10450093	Yes	No	No	Norm
CONFIGA	0FA0-0FAF	Input	8	10470100	Yes	No	No	
CONFIGB	0FB0-0FBF	Input	8	10470150	Yes	No	No	
CONFIGC	0FC0-0FCF	Input	8	104701A0	Yes	No	No	

VP Config

the name of the configuration used to open the database.

VP Database

the range of minidisk addresses that make up the database.

Status

"Input" and "Output" are shown for normal read/only and read/write databases. "Coupled" is shown for an output database that was opened with the "Coupled" option for loosely coupled testing. You can determine whether or not any other users have coupled to the same database by checking the "Users" count, or with VPQUERY USERS by key or configuration name.

Users

the number of VPARS users who are sharing this database.

DB Key

the database key. The key is an eight-character hexadecimal number used to uniquely identify each open database in the system. It is based on the real location of the first database minidisk.

Base

whether or not a TPF base system is associated with this configuration. "No" indicates that this is a "NOBASE" VPARS configuration. See "Testing Without a TPF Base System" on page 126 for more information.

Echo

whether or not TPF records read from the base system disks should be "echoed" to the read/write VPARS database.

ReqOL

Request-online; whether or not requests to read TPF records that are not present on the VPARS database should be passed to the RVTEXT user interface module. This module is generally used to request data records from an online system.

Clear

whether or not the database clear function is restricted. "Norm" means the clear function is normal, or not restricted; "Rest" means the clear function is restricted.

VPQUERY STATUS

Use the VPQUERY STATUS command to display a status summary containing VPARS database information, TPF total and per second I/O activity, and the values of various VPSET options.

VPQuery	Status	[CFg <i>cfgname</i>] [CONFig <i>cfgname</i>] [KEy <i>dbkey</i>] [LEvel <i>n</i>] [Userid <i>userid</i>]
---------	--------	---

CONFig *cfgname* (or) **CFg** *cfgname*

displays status for the database that was opened with the requested configuration name. If you do not also specify a userid, your open databases will be searched for the named configuration.

KEy *dbkey*

displays status for the single database with the requested key. The key is an eight-character hexadecimal number used to uniquely identify each open database in the system. It is based on the real location of the first database minidisk.

LEvel *n*

displays status for the database at the requested level in a multi-level database. This can be one of your open databases, or another user's database if you also specify the userid.

Userid *userid*

displays status for all open databases for the requested user. If you also specify a level number or a configuration name, you will see activity for one database.

Usage Notes:

1. VPQUERY entered without any sub-operands defaults to VPQUERY STATUS. Therefore, the minimum abbreviation for VPQUERY STATUS is VPQ.
2. The default is to display status for your primary open database. This is your first database, if you are using a multi-level database.
3. Key is mutually exclusive with any other option.
4. Level and Config are mutually exclusive.

Responses:

The VPQUERY STATUS command displays information by issuing several VPARS informational messages. The message numbers for VPQUERY STATUS are: B015I, B079I (optional), B080I (optional), B014I, B016I, B045I, B017I, B044I (optional), B019I, B020I, and B039I. Since these are informational messages, the message numbers will not appear. The optional messages may not appear. For example, message B080I shows the date the last checkpoint was set, and is not issued if no checkpoints have been set. See the Messages manual for a detailed description of each field. A sample response follows.

```
**** VPARS Status for database key 10450093 on 04/26/92 at 11:06:21 V3.3 ****
VPARS Base=0FF0-0FF0, Active=1, Full=14%, Sync=2927, Sync I/O=0
Blks=730, Actv=106, II=1, DI=1, RD=1, DB=2, Locked=0
VPARS I/O=7, Reads=6, Writes=1, Active=0, /sec=0
TPF records: Total=0, Added=0, Found=0, Updated=0
TPF Requests=1830, Active=1, /sec=2
Status Interval=200, Buffers: Minimum=25, In use=5, Maximum=2005
VPARS flags: PTV=off, Checkpoint=0 off, Clear=Normal
```

VPQUERY TPFSTATUS

Use the VPQUERY TPFSTATUS command to display TPF I/O activity for the current session. This is a subset of the information displayed by the VPQUERY STATUS command.

VPQuery	TPFstatus [ALL] [CFg <i>cfgname</i>] [CONFig <i>cfgname</i>] [KEy <i>dbkey</i>] [LEvel <i>n</i>] [Userid <i>userid</i>]
---------	---

ALL displays TPF status for all open VPARS databases.

CONFig *cfgname* (or) CFg *cfgname*

displays TPF status for the database that was opened with the specified configuration name. If you do not also specify a userid, your open databases will be searched for the named configuration.

KEy *dbkey*

displays TPF status for the single database with the specified key. The key is an eight-character hexadecimal number used to uniquely identify each open database in the system. It is based on the real location of the first database minidisk.

LEvel *n*

displays TPF status for the database at the specified level in a multi-level database. This can be one of your open databases, or another user's database if you also specify the userid.

Userid *userid*

displays TPF status for all open databases for the specified user. If you also specify a level number or a configuration name, you will see TPF status for one database.

Usage Notes:

1. The default is to display TPF status for your open databases.
2. Key is mutually exclusive with any other option.
3. Level and Config are mutually exclusive.

Response:

DB key	Users	TPF recs	Added	Found	Updated	Requests	Act	/sec
10470200	1	31250	1232	3340	904	136726	0	2

DB key

the database key. The key is an eight-character hexadecimal number used to uniquely identify each open database in the system. It is based on the real location of the first database minidisk.

Users

the number of users who currently have this database open.

TPF recs

the number of TPF records on the VPARS database.

Added

the number of TPF records that were added to the VPARS database since the database was opened.

Found

the number of TPF records that were read from the database since the database was opened. TPF read requests where the record was not found on the database, but returned from the base TPF system, are not included in this count.

Updated

the number of TPF records on the database that were updated (rewritten) since the database was opened.

Requests

the number of TPF I/O requests intercepted by VPARS since the database was opened.

Active

the current number of outstanding TPF I/O requests.

/sec per second; the number of TPF I/O requests issued during the previous second.

VPQUERY USERS

Use the VPQUERY Users command to display the userids, configuration names, device table names, and database key for some or all open VPARS databases.

VPQuery	Users [ALL] [Cfg <i>cfgname</i>] [CONFig <i>cfgname</i>] [KEy <i>dbkey</i>] [LEvel <i>n</i>] [Userid <i>userid</i>]
---------	---

ALL displays userids, keys, configuration, and device table names for all open databases.

CONFig *cfgname* (or) Cfg *cfgname*

displays userids, keys, configuration, and device table names for all databases open with the given configuration name.

KEy *dbkey*

displays userids, keys, configuration, and device table names for all users who have the database with the given key open. The key is an eight-character hexadecimal number used to uniquely identify each open database in the system. It is based on the real location of the first database minidisk.

LEvel *n*

displays userids, keys, configuration, and device table names for all users who have the database open which you have open at level *n*. If you also supply a userid, the database that user has open at level *n* is checked, and all users who have the same database open are shown.

Userid *userid*

displays userids, keys, configuration, and device table names for all users who have the database open that the given user has open. If you also supply a level number, the database that user has open at that level is checked, and all users who have the same database open are shown.

Usage Notes:

1. The VPQUERY USERS command is primarily intended to show which users have the database with a given key open. For example, if you need to update a database that is used as an intermediate database level by several users, this command will tell you who has the database open. You can then warn these users that modifications are being made to the database.

Response:

DB key	Userid	Config	DevTbl	Userid	Config	DevTbl
10470200	VPTPF1	CONFIG1	DEVTBL1	VPTPF2	SPECTST	SPECTBL

DB key

the database key. If you are displaying information for a userid or 'ALL', several lines may be shown. The list of users who have the database with the given key open may span one or more output lines, but each line will pertain to only one key.

Userid

the userid of a user who has the database open.

Config

the configuration name that the user used to open the database.

DevTbl

the device table name that the user used at open time.

VPQUERY VPSTATUS

Use the VPQUERY VPSTATUS command to display VPARS database status for the current session. This is a subset of the information displayed by the VPQUERY STATUS command.

VPQuery	VPstatus [ALL] [CFg <i>cfgname</i>] [CONFig <i>cfgname</i>] [KEy <i>dbkey</i>] [LEvel <i>n</i>] [Userid <i>userid</i>]
---------	--

ALL displays VPARS status for all open databases.

CONFig *cfgname* (or) CFg *cfgname*

displays VPARS status for the database that was opened with the requested configuration name. If you do not also specify a userid, your open databases will be searched for the named configuration.

KEy *dbkey*

displays VPARS status for the single database with the requested key. The key is an eight-character hexadecimal number used to uniquely identify each open database in the system. It is based on the real location of the first database minidisk.

LEvel *n*

displays VPARS status for the database at the requested level in a multi-level database. This can be one of your open databases, or another user's database if you also specify the userid.

Userid *userid*

displays VPARS status for all open databases for the requested user. If you also specify a level number or a configuration name, you will see VPARS status for one database.

Usage Notes:

1. The default is to display VPARS status for your open databases.
2. Key is mutually exclusive with any other option.
3. Level and Config are mutually exclusive.

Response:

DB key	Users	VPARS	I/O	Reads	Act	/sec	Blks used	RD	DB	Full
10450093	1		8	6	0	0	106	1	2	14%

DB key

the VPARS database key. The key is an eight-character hexadecimal number used to uniquely identify each open database in the system. It is based on the real location of the first database minidisk.

Users

the number of users who currently have this database open.

VPARS I/O

the number of reads and writes performed by VPARS to the VPARS database.

Reads

the number of reads performed by VPARS to the VPARS database.

Act the number of I/O requests to the VPARS database that are currently active (in progress).

/sec per second; the number of I/O requests performed against the VPARS database in the previous second.

Blks used

the total number of 4096-byte blocks in use in the database.

RD the number of record directory blocks in use in the database.

DB the number of data blocks in use in the database.

Full the percent full of the database.

VPREL

Use the VPREL command to release the private VPARS mini-disk, associated with your userid or privilege classes, from CP control.

VPREL	
-------	--

Usage Notes:

1. What minidisk to release is determined by the PARM_USER and/or PARM_CLASS keywords in the VPSYSTEM DEFAULTS file.
2. Care should be taken when releasing the disk when VPARS is active. Although, once initialized, VPARS needs not refer to the files on disk, any VTINIT (configuration changes) and some query commands do require access to the VPARS files.

VPREMOVE

Use the VPREMOVE command to remove inactive minidisks from your VPARS database.

VPREMove	<i>n</i> <i>-n</i>
----------	---------------------------

- n* specifies the number of inactive disks to be removed from the database. The default is 1. If the number specified is greater than the number of inactive disks, all inactive disks will be removed.
- n* specifies that all inactive disks, minus the number specified, should be removed from the database. If minus zero (-0) is specified all inactive disks are removed. If the number of inactive disks is less than the number specified, no disks are removed.

Usage Notes:

1. Active disks (VPARS database disks with any TPF records on them) will not be removed. If you need to detach an active database disk, close your database and use the CP DETACH command.

Response:

VPARS disk *vdev* has been removed from the database

VPSET

The VPSET command sets various VPARS options. Each subcommand is described separately on the following pages.

VPSet	<pre>{ BASE { Yes No } } { BUild { options } } { CHeckpoint [OFF] } { CHkpoint [OFF] } { CLear { NORMAl RESTRicted } } { COupled { Yes No } } { ECHO { Yes No } } { INTercept { ON OFF [vdev ...] } } { MAXOnq n [LEvel n] } { MAXUsers n { USerid userid KEy dbkey } } { MINOnq n [LEvel n] } { PTV { ON OFF } } { REQOl { Yes No } } { REQOnline { Yes No } } { STATustime n } { UNSUPCcw { ON OFF } } { UNSUPMsg { ON OFF } }</pre>
-------	--

Usage Notes:

1. (ON and Yes) and (OFF and No) are synonymous; either set can be used for all VPSET commands except VPSET BUILD.

VPSET BASE

Use the VPSET BASE command to cause VPARS to not issue any reads to the base TPF system.

VPSET	BASE	{ Yes No }
-------	------	--------------

Yes tells VPARS that this is a "normal" database, and records that are read by the application and are not found on the VPARS database are read from the base TPF disks.

No makes this a NOBASE system, in which records that are not found on the VPARS database are not read from the base disks. Instead, a record consisting of binary zeros is created by VPARS and returned to the application.

Usage Notes:

1. IPL text may still be read from the base disks in a NOBASE system, if the application reads IPL text and it is not present on the VPARS database.

Responses:

Requested option has been set

The database has been placed in BASE or NOBASE mode as requested.

VPSET BUILD

Use the VPSET BUILD command to set VPARS up to build a NOBASE VPARS database.

VPSET	Build { ALL { ON OFF } } { DIrcomp { ON OFF } } { NOupdate { ON OFF } } { SYnctime <i>n</i> } options: KEy <i>dbkey</i> LEvel <i>n</i>	<i>options</i>
-------	--	----------------

ALL { ON | OFF }

sets all build options on or off and sets sync time to 60 for on and 2 for off.

DIrcomp { ON | OFF }

ON causes the VPARS record directory and indexes to be compacted. This option should only be used when building a NOBASE system and TPF records are being written to disk sequentially by track address.

KEy *dbkey*

specifies the key of the open database to be set. If you are attempting to set a value for another user's database, you must have the set for another command privilege class.

LEvel *n*

specifies the level of a multi-level database for which the MINONQ value should be set. If not specified, the setting applies to the first level (generally the read-write database).

NOupdate { ON | OFF }

ON suppresses updating of records found on the VPARS database. This option improves performance when a restart must be done on a build function. Build implies record adds only.

SYnctime *n*

specifies the time interval between directory synchronizations in seconds. The minimum is 2 and the maximum is 60. The normal directory synchronization interval for a VPARS database is 2 seconds.

Usage Notes:

1. A NOBASE VPARS database contains all of the TPF records that are required to run TPF. A base TPF system is not required.
2. The VPSET BUILD option was designed to improve performance and reduce disk utilization when building a NOBASE VPARS database. Performance will be greatly reduced if these options are used for normal VPARS operation. Also, records that exist on the database will not be updated if the NOUPDATE option is specified; this will cause erroneous results if used during normal TPF testing.

Responses:

Requested option has been set

VPSET CHECKPOINT

Use the VPSET CHECKPOINT (or VPSET CHKPOINT) command to set a database checkpoint. You can open or clear a database to any previously set checkpoint. This restores the database to the records that were active when the checkpoint was set.

VPSET	{ CHkpoint } [Off] { CHeckpoint }
-------	--

Off turns off the checkpointing function. See the usage notes for more information. If you do not specify OFF, a checkpoint is set.

Usage Notes:

1. A checkpoint can be set at any time, and any number of checkpoints can be set.
2. The database can be cleared to a checkpoint with VPCLEAR CHECKPOINT or VPCLEAR CHKPOINT.
3. The database can be opened to a checkpoint with VPOPEN CHECKPOINT or VPOPEN CHKPOINT.
4. When a checkpoint is set, any record on the VPARS database that is updated will be written to a new location on the database. This allows the old version of the record to be reinstated when the database is cleared to a checkpoint. However, this may require a lot of space on the database.
5. During a test, if you determine that you will not need to clear the database to any previously set checkpoint, turning the checkpointing function off allows previously existing records to be updated in place. If you turn checkpointing off, you will not be able to clear or open to any checkpoint that was set.

Responses:

VPARS checkpoint *nn* set

A checkpoint has been taken as requested.

Requested option has been set

The checkpointing function has been turned off as requested.

VPSET CLEAR

Use the VPSET CLEAR to set the clear function to normal or restricted.

VPSET	CLear	{ NORMal } { RESTRicted }
-------	-------	------------------------------

NORMal

places no restrictions on any VPCLEAR, VPIPL or VPOPEN clear operations.

RESTRicted

requires the configuration name of the database being cleared to be entered for VPCLEAR and VPIPL CLEAR, and for VPOPEN CLEAR and VPOPEN CHECKPOINT commands.

VPSET COUPLED

Use the VPSET COUPLED command to allow other users to open your VPARS database for loosely-coupled testing.

VPSET	COupled	{ Yes No }
-------	---------	--------------

Yes allows other users to "join" your open VPARS database for loosely-coupled testing. In coupled mode, all users write to the same database.

No prevents other users from joining your open database. There must be no other users coupled to your database.

Usage Notes:

1. In a loosely-coupled database, record locking is the responsibility of the application. VPARS does not provide record locking.

Responses:

Requested option has been set

The database has been set to allow (or not allow) other users to open it in loosely-coupled mode.

VPSET ECHO

Use the VPSET ECHO command to cause a copy of all records that are read from the base TPF system to be rewritten to the VPARS database.

VPSET	ECHO	{ Yes No }
-------	------	--------------

Yes starts the "read-echo" function, which causes all records that VPARS reads from the base TPF system to be rewritten to the VPARS database.

No stops the read-echo function. Records read from the base TPF system are no longer rewritten to the VPARS database.

Usage Notes:

1. The ECHO function can be useful in building a VPARS database that can then be used in a NOBASE system (without an underlying set of TPF disks).

Responses:

Requested option has been set

The read-echo function has been turned on or off as requested.

VPSET INTERCEPT

Use the VPSET INTERCEPT command to turn VPARS interception of I/O to disks in the TPF base system ON or OFF.

VPSET	INTercept	{ ON } { OFF [<i>vdev vdev vdev-vdev ...</i>] }
-------	-----------	--

ON flags all TPF disks to have I/O intercepted by VPARS. All disks in the TPF device range (as defined by the TPF device table selected during VPOPEN) will have their I/O intercepted.

OFF [*vdev vdev vdev-vdev ...*]
removes the VPARS intercept flag from the specified TPF disks. I/O to these disks will no longer be intercepted by VPARS. A list of virtual devices (separated by blanks), or a range of devices (consisting of a starting and ending device number, separated by a dash) will remove the intercept flag from each device in the list or range. If a list or range of devices is not specified, VPARS interception is turned off for all TPF devices.

Usage Notes:

1. When VPARS interception is turned on, the disks are also switched to read/write; otherwise, VM's CCW translation function would reject write CCWs as invalid. However, when VPARS is intercepting the I/O, all write CCWs are handled by VPARS, and the records are actually written to the VPARS database.
2. TPF disks that are being intercepted by VPARS are flagged as VPARS in the output from the CP Query Virtual Dasd command.
3. TPF disks should be linked read/only before using VPOPEN or VPSET INTERCEPT ON. If the disks are linked read/write, VPOPEN and VPSET INTERCEPT ON will not flag them to be intercepted. If you link TPF disks read/write, be sure that you really want to write on them.

Responses:

```
I/O intercept set { on | off } [ for vdev ]
```

I/O interception has been set on or off as requested.

VPSET MAXONQ

Use the VPSET MAXONQ to set the maximum number of I/O tasks that can be active on the I/O queues of a VPARS database.

VPSET	MAXONq <i>nnnn</i> [L E vel <i>n</i>] [K E y <i>dbkey</i>]
-------	---

nnnn

specifies the maximum number of active I/O tasks. The minimum value for maxonq is 512, and the maximum number is 4096. The minimum spread between minonq and maxonq is 255.

K~~E~~y *dbkey*

specifies the key of the open database to be set. If you are attempting to set a value for another user's database, you must have the set for another command privilege class.

L~~E~~vel *n*

specifies the level of a multi-level database for which the MAXONQ value should be set. If not specified, the setting applies to the first level (generally the read-write database).

Usage Notes:

1. Setting the maxonq value can severely impact the performance of a VPARS database and the VM system. The initial value for maxonq is set by a value in the VPARS configuration used for the open. We recommend that you leave these values set as they are in the distributed code.

Responses:

Requested option has been set.

VPSET MAXUSERS

Use the VPSET MAXUSERS command to limit the number of users who can open a VPARS database. This is normally a class B or P command.

VPSET	MAXUsers <i>n</i> [KEy <i>dbkey</i>] [USerid <i>userid</i>]
-------	--

n specifies the maximum number of VPARS databases that can be open systemwide. When this limit is reached, additional VPOPEN attempts will fail. You can display the current limit and the current number of VPARS users with VPQUERY MAXUSERS.

KEy *dbkey*

specifies the key of an open database and allows you to restrict the number of users on a specific VPARS database. You must have the set for another command privilege class.

USerid *userid*

specifies the userid of a guest that has the VPARS database open for output. Allows you to restrict the number of users on a specific VPARS database. You must have the set for another command privilege class.

Responses:

Maximum number of VPARS users set to *n*. Currently *n* active users.

The maximum number of VPARS users has been set to the requested value.

VPSET MINONQ

Use the VPSET MINONQ to set the minimum number of I/O tasks maintained on the I/O queues of a VPARS database.

VPSET	MINONq <i>nnnn</i> [L E vel <i>n</i>] [K E y <i>dbkey</i>]
-------	---

nnnn

specifies the minimum number of active I/O tasks. The minimum value for minonq is 5, and the maximum number is 2048. The minimum spread between minonq and maxonq is 255.

K~~E~~y *dbkey*

specifies the key of the open database to be set. If you are attempting to set a value for another user's database, you must have the set for another command privilege class.

L~~E~~vel *n*

specifies the level of a multi-level database for which the MINONQ value should be set. If not specified, the setting applies to the first level (generally the read-write database).

Usage Notes:

1. Setting the minonq value can severely impact the performance of a VPARS database and the VM system. The initial value for minonq is set by a value in the VPARS configuration used for the open. We recommend that you leave these values set as they are in the distributed code.

Responses:

Requested option has been set.

VPSET PTV

Use the VPSET PTV command to turn PTV mode on or off.

VPSET	PTV	{ ON OFF }
-------	-----	--------------

ON or OFF

turns PTV mode on or off. PTV mode is a special mode of VPARS operation, which is described in “PTV Operation” on page 20.

Usage Notes:

1. When you turn PTV mode off, your database will be cleared.

Responses:

PTV mode entered

PTV mode has been turned on as requested.

VPARS database clear complete

PTV mode has been turned off as requested, and the database has been cleared.

VPSET REQONLINE

Use the VPSET REQONLINE (or VPSET REQOL) command to cause VPARS to call a user interface module to request records from an online system, when the application reads a record that is not found on the VPARS database.

VPSET	REQOnline	{ Yes No }
	REQOl	{ Yes No }

Yes places the database into "request-online" mode, in which all records that are read by the application and not found on the database are passed to a user-interface CP module, which can request the records from an online system (or any other source).

No removes the database from "request-online" mode.

Usage Notes:

1. VPARS is distributed with a prototype user-interface module, called RVTEXT. You can modify this module to provide the requested record, and assemble it into your VM system. Entry point RVTEXTOL is called to request a record that VPARS did not find on the database. See the module prologue in RVTEXT for details on the interface parameters.
2. When REQONLINE is set to YES, VPARS also sets BASE to NO.

Responses:

Requested option has been set

The request-online function has been turned on or off as requested.

VPSET STATUSTIME

Use the VPSET STATUSTIME command to set the time interval between automatic VPQUERY STATUS displays.

VPSET	Statustime <i>n</i>
-------	---------------------

n sets the number of minutes between automatic VPQUERY STATUS displays. Specify a number between zero and 65535. If zero is specified, no automatic status displays are done.

Response:

Status interval set to *n* minutes

Status will be automatically displayed at the requested intervals.

VPSET UNSUPCCW

Use the VPSET UNSUPCCW command to tell VPARS whether or not you want CCW chains displayed for unsupported read CCWs.

VPSET	UNSUPCcw { ON OFF }
-------	-----------------------

ON | OFF

On causes CCWs to be displayed for read CCWs that VPARS does not support. This is the default when the database is opened. Off causes these CCWs not to be displayed.

Usage Notes:

1. When VPARS encounters a read CCW that it does not support, the I/O will be rejected with a COMMAND REJECT. The VPSET UNSUPCCW command controls whether or not the CCW is displayed on the console. The related VPSET UNSUPMSG command controls whether or not a message is issued.
2. If you receive any "unsupported CCW" messages, you should report them to Virtual Software Systems. If they are valid CCWs, VPARS will be updated to support them.

VPSET UNSUPMSG

Use the VPSET UNSUPMSG command to tell VPARS whether or not you want a message issued for unsupported read CCWs.

VPSET	UNSUPMsg { ON OFF }
-------	-----------------------

ON | OFF

On causes a message to be issued when VPARS encounters a read CCW chain that it does not support. On is the default when the database is opened. Off suppresses these messages.

Usage Notes:

1. When VPARS encounters a read CCW that it does not support, the I/O will be rejected with a COMMAND REJECT. The VPSET UNSUPMSG command controls whether or not a message is issued. The related VPSET UNSUPCCW command controls whether or not the CCW is also displayed.
2. If you receive any "unsupported CCW" messages, you should report them to Virtual Software Systems. If they are valid CCWs, VPARS will be updated to support them.

VPTEST

Use the VPTEST command to check the syntax of the VPARS configuration files.

VPTest	<i>fname ftype</i>
--------	--------------------

fname

The name of the file to check.

ftype One of the following, to identify the type of file.

- DEFAULTS for VPSYSTEM files
- VPCONFIG for configuration files
- VPCONCAT for concatenation files
- VPTPFDEV for device table files
- VPTPFFMT for format table files

The file being tested will go through syntax checking, but it will not be put online even if the test is successful.

Chapter 7. VPARS CMS Commands

This section describes the VPARS CMS commands. The format, use, and the normal responses of each command are explained.

VPBKUP

Use the VPBKUP command to back up all TPF records on your VPARS database to a standard labeled tape.

VPBKUP	<i>vdev altvdev</i> (options options: CFg <i>cfgname</i> OPer <i>userid</i> COnfig <i>cfgname</i> REtPd <i>days</i> DEn <i>density</i> VPvdev <i>vdev</i> MOde <i>xx</i> VTape
--------	--

vdev a tape device number. The default is 181. You may use an asterisk to default the first drive to 181, and specify an alternate. You can use any valid CMS tape device number (180-187 and 288-28F).

altvdev

an alternate tape device number. If the backup takes more than one tape, VPBKUP will switch to the alternate drive after the first tape is full. If the second tape fills, VPBKUP will switch back to the primary drive. If you do not specify an alternate, VPBKUP will only use one drive.

CFg *cfgname* (or) **COnfig** *cfgname*

a VPARS configuration name. If you specify a name, the database mini-disks for the named configuration will be backed up. If you do not specify a name, your default database will be backed up.

DEn *density*

the requested tape density.

MOde *xx*

a tape mode set byte. *xx* is the hexadecimal representation of a mode set byte. DEN and MODE are mutually exclusive.

OPer *userid*

specifies that you want a copy of tape-related messages to be sent to another userid, such as the system operator or a tape operator.

REtPd *n*

the number of days you want the backup tape kept. The value you supply is used to obtain an expiration date, which is written to the standard label of the output tape.

VPvdev vdev

the starting virtual device number for the input VPARS database. You should have a read link to all minidisks in the input database. If this parameter is not specified, your normal VPARS database will be backed up.

VTape

automatically mounts virtual scratch tapes to satisfy mount requests if you are taking the backup to virtual tapes.

Usage Notes:

1. VPBKUP calls an external exec, VSSLTAPE, passing it all tape mount-related messages it issues. You can customize this exec to interface to a tape management system. See the description of the VSSLTAPE exec for more information.
2. You must have a tape drive at virtual device 181. This is checked after the VSSLTAPE exec is passed the Mount message (2301A).
3. If you use the VTAPE option, you should define a virtual tape drive at 181 before calling VPBKUP. In this case, VPBKUP will perform the VTAPE mounts; the VSSLTAPE exec does not need to.
4. The VPBKUP command writes all active data, checkpoints, and IPL text from your VPARS database to tape. The database must be closed during the backup. You can specify a different database (other than your default) with the VPVDEV option or the CONFIG (or CFG) option.
5. When the data is restored with VPREST, you must have the same number of minidisks linked as you had during the VPBKUP. The minidisks do not have to be the same device type, but they must be able to hold all the active data blocks that were backed up.
6. Two files are created on your A-disk during a backup: a cycle file that contains the volume serials of the tapes created, and a control file that contains the date, time, and cycle number of the current backup. These files are used by VPREST to request input tapes for a restore.

The filetype of the cycle file is VPBKUP CYCLnnnn, where nnnn is the cycle number. The filetype of the control file is NCONTROL during the backup. If the backup completes normally, the current control file is renamed to a filetype of OCONTROL and the new control file is rename to a filetype of CONTROL.

7. If neither DEN nor MODE is specified, the density is set to the highest available for the type of tape drive.
8. Use the DEN parameter if you want a density other than 6250 on a 9-track tape drive.
9. If you specify an operator userid, a copy of all tape messages will be sent to the designated user. The tape drive address in the messages will be your

virtual tape drive, and your userid will be in the message to help the operator relate your virtual drive address to the real address. The operator will receive a copy of all messages related to tape processing, including those requiring a reply, but you must enter the replies.

Responses:

Backing up *vdev*

Issued for each virtual disk processed.

VPARS backup complete, *nn* blocks backed up

Issued after all active blocks have been backed up.

VPBLDFMT

Use the VPBLDFMT command to read a formatted TPF disk and create a text file containing a device format table.

VPBLDFMT	<i>vdev</i>
----------	-------------

vdev the device number of a formatted TPF disk that you have linked.

Usage Notes:

1. VPBLDFMT will produce a text file on the A-disk. The filename of the text file matches the volume label of the TPF disk that it read to build the format table. You may want to move this text file to your VPARS installation disk, and you can rename it if necessary.
2. If the format of a TPF disk changes, you should rebuild the format tables.
save
3. You can move the resulting format table text file to the VSSI parm disk.
4. You will receive messages on your terminal showing starting and ending cylinders, tracks, and TPF record lengths. These messages are produced for documentation purposes; the text deck produced by VPBLDFMT is ready to be moved to the VSSI parm disk.
5. VPBLDFMT is intended to be run against a completely formatted TPF disk. If all cylinders of the disk have not been formatted by the TPF formatter, VPBLDFMT may not be able to read the first count field on a track, and you may receive I/O errors.
6. The VPBLDFMT command will take a few minutes to run.

VPBXBMAP

Use the VPBXBMAP command to copy pool directory records from tape to a CMS disk, for use by VPBXREST.

VPBXBMAP	<i>ctln</i> (options options: DEnsity <i>density</i> MOde <i>xx</i>
----------	---

ctln is the filename of the control file. The filetype for this file is VPBMCTL. This file contains the control information for processing.

DEnsity *density*
the requested tape density.

MOde *xx*
a tape mode set byte. *xx* is the hexadecimal representation of a mode set byte. DEN and MODE are mutually exclusive.

Usage Notes:

1. VPBXBMAP reads a tape(s) created by a TPF online utility that contain the long term pool directory bit map records. The record headers are removed and the 1000 byte bit maps are written to a CMS file
2. The long term pool directory bit map records must be on the tape(s) in record allocation order.
3. A sample VPBMCTL file is distributed with VPARS. The filename for this file is TPF24MOD.
4. For a description of the VPARS NOBASE feature, see “Testing Without a TPF Base System” on page 126.
5. VPBXBMAP calls an external exec, VSSLTAPE, passing it all tape mount-related messages it issues. You can customize this exec to interface to a tape management system. See the description of the VSSLTAPE exec for more information.

Sample VPBXBMAP control file:

```
TPF24MOD 183 706828 DIR.TAPE  
TPF24MOD 184 134678 DIR.TAPE
```

Records are read from the VPBXBMAP VPBMCTL file when a VPBXBMAP is started.

If more than one tape is required (multi volume dataset) multiple records can be entered, with the following requirement: Although all parameters must be present for each record, only parm 2 (tape address) and parm 3 (volser) are considered for records 2 thru n. Parm 1 and 4 are kept from the first record read.

The first entry on the record is the filename for the pool directory bit map file. The filetype will be VPBXRMAP

The second entry is the VDEV number of the tape drive that the online pool bit map directory tape is mounted on.

The third entry is the volume serial of the tape.

The fourth entry is the dataset name of the tape.

VPBXPLTB

Use the VPBXPLTB command to build a TPF pool section directory map.

VPBXPLTB	<i>fname fmode</i>
----------	--------------------

fname

is the CMS filename for the pool section directory.

fmode

is the CMS filemode for the pool section directory.

Usage Notes:

1. VPBXPLTB is assembled against the TPF system definition macros SYCON, CZ1GF and CLHEQ. When it is executed it writes a pool section directory to a CMS file. This file is used with the pool directory bit map during a VPBXREST restore to determine if a pool record is in use. If the record is not in use, it is not restored to the database.
2. See the VPBXPLTB exec description for assembly against the TPF macro libraries.
3. For a description of the VPARS NOBASE feature, see “Testing Without a TPF Base System” on page 126.

VPBXREST

Use the VPBXREST command to restore records from a TPF capture (BXA) tape.

VPBXREST	<i>ctlfn</i> (options
	options:
	ALLrecs MODe <i>xx</i>
	DEnsity <i>density</i> STart <i>cchh</i>
	ENd <i>cchh</i> STerm
	FReczero TApes <i>nn</i>
	LTerm ZERo

ctlfn is the restore control file name. The filetype for this file is VPBXRCTL. This file contains the filenames for all other files used by the restore and the dataset name of the BXA capture tape.

ALLrecs

restore all records on capture tape.

DEnsity *density*

the requested tape density.

ENd *cchh*

is the ending cylinder and head number in hex if only a range of tracks is being restored.

FReczero

Specify that the entire fixed record must be zero, to be recognized as a null record. If this option is not specified, only the record ID is checked for zero.

LTerm

restore all long term pool records.

MODe *xx*

a tape mode set byte. *xx* is the hexadecimal representation of a mode set byte. DEN and MODE are mutually exclusive.

STart *cchh*

is the starting cylinder and head number in hex if only a range of tracks is being restored.

STerm

restore all short term pool records.

TApes *nn*

is the number of tapes to be processed.

ZERO

restore records the are all zero.

Usage Notes:

1. VPBXREST provides the ability to restore all or selected records from a TPF capture tape. It was written to support building a VPARS NOBASE database. NOBASE is a VPARS feature that allows TPF testing without a base system. For a description of the VPARS NOBASE feature, see “Testing Without a TPF Base System” on page 126.
2. Sample files are distributed with VPARS for all of the files used by VPBXREST. The filename for these files is TPF24MOD.
3. VPBXREST requires a tape drive at address 181.
4. VPBXREST calls an external exec, VSSLTAPE, passing it all tape mount-related messages it issues. You can customize this exec to interface to a tape management system. See the description of the VSSLTAPE exec for more information.

Sample VPBXREST control file:

```

TPF24MOD TPF24MOD TPF24MOD TPF24M01 TPF24MOD TPF24MOD BXA.TAPE
VPBXRCFG VPBXDIR VPBXRFMT VPBXRVOL VPBXREXR VPBXREXI CAPT DSN

```

One record is read from the VPBRXCTL control file when a restore is started.

The second record in this file is the required filetypes of the files containing restore information. Only the first record in this file is read by the restore program.

The first entry on the record is the filename of the TPF disk configuration file. The entries in the disk configuration file relate a TPF module number to a virtual device number and a disk volume serial.

The second entry on the record is the filename of the TPF pool directory bit map file. The records in the bit map file are the 1000 byte bit map records built by the VPBXBMAP module from a capture of the pool directories.

The third entry on the record is the filename of the pool format file. Each record in the pool format file identifies the extents of a pool area on the TPF modules.

The fourth entry on the record is the capture tape volume serial file. The volume serial file identifies the tapes that are to be processed for this restore. The filename of this file is different from the others because it is assumed that multiple virtual machines will be used for the restore, but the restore control files will be maintained on a minidisk that is shared by all machines. If the TAPES option of VPBXREST is used, this file is not used.

The fifth entry on the record is the filename the track range exclude file.

The sixth entry on the record is the filename of the TPF record ID exclude file.

The seventh entry on the record is the dataset name on the TPF capture tapes.

Responses:

TPF system date and time from capture tape *date time*

Restoring module *module* tape *nn* to *vdev volser* tape *volser volser*

The following cylinder and head ranges will not be restored

From CCHH *cchh* to CCHH *cchh*

TPF records with the following record IDs will not be restored

id id id

nn records processed from tape *volser* for module *module* on *vdev*

nn records restored, *nn* zero records were not restored

nn Long term pool records were restored

VPFMT

Use the VPFMT command to format a minidisk for use in a VPARS database.

VPFMT	<i>vdev</i> [<i>volser</i>] ([REFormat] [REStart <i>hexcyl</i>])
-------	--

vdev

specifies the virtual device number (address) of a minidisk to be formatted for use by VPARS.

volser

specifies an optional volume serial to be used in the volume label. The default volume serial is VP*nnnn*, where *nnnn* is the virtual device number of the minidisk.

REFormat

specifies that the entire database minidisk has already been formatted by VPARS. The REFORMAT option rewrites only the header and control records, and is therefore much faster.

REStart *hexcyl*

specifies the cylinder number for a restart, in hexadecimal. If a VPFMT fails with an I/O error, you can restart VPFMT after correcting the problem.

Usage Notes:

1. You must use VPFMT to format a minidisk before it can be used in a VPARS database. The database must be closed during formatting.
2. Do not specify REFORMAT if you have moved or enlarged a VPARS database minidisk, or are formatting a newly allocated minidisk; the entire minidisk must have already been formatted with VPFMT.

Responses:

Format disk on *vdev*? Enter YES or NO

This is a request to verify the device to be formatted. If the device number is correct, enter YES; if not, enter NO.

Formatting minidisk *vdev*

This response is issued when formatting begins.

n cylinders formatted on *vdev volser*

This response is issued when formatting is complete.

VPIPLWR

Use the VPIPLWR command to write IPL bootstrap and text records to track zero of a disk. The IPL records can be written in TPF or Device Support Facilities (DSF/OS) format.

VPIPLWR	<i>vdev</i> [<i>fname</i> [<i>iplboot</i>]]
---------	---

vdev is the virtual device number of the TPF disk to receive the IPL text.

fname

is the filename of the CMS file containing the IPL text. The filetype of the file must be TEXT. The default filename is IPLTEXT.

iplboot

is the filename of a text file that contains IPL bootstrap records 1 and 2. The filetype of the file must be TEXT. The default filename is IPLBOOT.

Usage Notes:

1. DSF/OS format IPL bootstrap consists of standard IPL records 1 and 2 and a single IPL record as record 4 on track zero. TPF format IPL bootstrap consists of special IPL record 1 and 2 that will process multiple IPL records starting with record 4 on track zero.
2. There are two IPL bootstrap sample assemble files on the VPARS distribution disk (IPLBOOT and TPFBOOT). IPLBOOT contains IPL records 1 and 2 for a DSF/OS format IPL. TPFBOOT contains IPL records 1 and 2 for a TPF format IPL. A prolog in each of the sample files describes their use and format.
3. If you use the IPLBOOT file, VPIPLWR will load the IPLBOOT and IPL text files, update the read CCW in IPL record 2 for the length of the IPL text record, read the label from the *vdev* and write records 1, 2, 3 and 4.
4. If you use the TPFBOOT file, VPIPLWR will load the IPLBOOT and IPL text files, read the label from the *vdev*, write records 1, 2, 3 and use the lengths in the read CCWs of IPL record 2 to segment the IPL text into multiple records starting with record 4. The IPL record 2 in this file also supports the record cache RPQ.
5. The TPFBOOT file is designed to copy in the assembler source for IPL records 1 and 2. The sample TPFBOOT will write three X'FF8' byte records for IPL text. The copy file for the TPFBOOT sample file is TPFIPLR2 COPY. You can replace this copy file with your IPLA source, or you can update the copy file to do any special processing required by your IPLB.

6. If the VPARS database is open, VPARS will intercept the I/O, and the IPL text will be written to VPARS. However, you should always specify the device number of the TPF disk for *vdev*.
7. If the VPARS database is closed, and you have a write link to the TPF disk, VPIPLWR will write the IPL text to the TPF disk.

Responses:

IPL text initialized on *vdev*

The IPL text has been written as requested.

VPLOAD

Use the VPLOAD command to restore TPF records from a VPUNLD logical backup tape to your VPARS database.

VPLOAD	<i>vdev altvdev</i> (options options: CYcle <i>nn</i> OPer <i>userid</i> DEn <i>density</i> TApes <i>n</i> MOde <i>xx</i> ULtape <i>volser</i> MAp VTape
--------	---

vdev a tape device number. The default is 181. An asterisk can be used to default the first drive to 181 and specify an alternate. You can use any valid CMS tape device number (180-187 and 288-28F).

altvdev

an alternate tape device number. If the unload took more than one tape, VPLOAD will switch to the alternate drive after the first tape is completely read. After reading the second tape, VPLOAD will switch back to the primary drive. If you do not specify an alternate, VPLOAD will only use one drive.

CYcle *nn*

requests that a specific cycle file be used to identify the input tape volumes for this load.

DEn *density*

a requested tape density.

MOde *xx*

a tape mode set byte. *xx* is the hexadecimal representation of a mode set byte. DEN and MODE are mutually exclusive.

MAp

issues a message showing total TPF records unloaded and total VPARS database blocks in use when the VPUNLD tape was created.

OPer *userid*

sends a copy of all tape messages to an operator userid.

ULTape *volser*

requests a specific tape volume for the load. At the end of the volume, VPLOAD will ask for the volume serial of the next tape to be used.

TApes *n*

requests processing of a specific number of tapes with unspecified volume serial numbers and dataset names. It is the users responsibility to mount the correct unload tapes in the correct order.

VTape

automatically mounts virtual input tapes to satisfy mount requests, if you are loading the data from virtual tapes. If VTAPE is specified and the tape drive is a real device, or a volume is already mounted on the drive, an error message is issued and the restore terminates.

Usage Notes:

1. VPLOAD calls an external exec, VSSLTAPE, passing it all tape mount-related messages it issues. You can customize this exec to interface to a tape management system. See the description of the VSSLTAPE exec for more information.
2. You must have a tape drive at virtual device 181. This is checked after the VSSLTAPE exec is passed the Mount message (2301A).
3. If you use the VTAPE option, you should define a virtual tape drive at 181 before calling VPLOAD. In this case, VPLOAD will perform the VTAPE mounts; the VSSLTAPE exec does not need to.
4. If neither ULTAPE nor CYCLE is specified, the current cycle in the control file will be used. If control and cycle files do not exist, VPBKUP will issue an error message.
5. You can look at the cycle file to determine the tape volumes that will be required for a load.
6. Your VPARS database can be open or closed during a load. The load is performed by writing the TPF records to their corresponding TPF record address. If your VPARS database is open, VPARS will intercept these writes and place the records on your VPARS database. If your database is closed, and you have write links to a base TPF system, the records will be loaded to the base TPF system.
7. If an operator userid is specified, a copy of all tape messages will be sent to the designated user. The tape drive address in the messages will be your virtual tape drive, and your userid will be in the message to help the operator relate your virtual drive address to the real address. The operator will receive a copy of all messages related to tape processing, including those requiring a reply, but you must enter the replies.

Responses:

Unload tape created on *date* at *time*
Load TPF records? - Enter YES or NO.

This verifies the request to load TPF records into your VPARS database or TPF base system. If the records should be loaded, enter YES, if not, enter NO.

Keep 181 SL *pnnnnn* dsname=*dsname*

This response is issued after all TPF records have been loaded.

VPREST

Use the VPREST command to restore a VPBKUP backup tape to your VPARS database.

VPREST	<i>vdev altvdev</i> (options options: BKtape <i>volser</i> MOde <i>xx</i> CFg <i>name</i> OPer <i>userid</i> COntig <i>name</i> TApes <i>n</i> CYcle <i>nn</i> VPvdev <i>vdev</i> DEn <i>density</i> VTape MAp
--------	--

vdev a tape device number. The default is 181. An asterisk can be used to default the first drive to 181 and specify an alternate. Any valid CMS tape device number can be used (180-187 and 288-28F).

altvdev

an alternate tape device number. If the backup took more than one tape, VPREST will switch to the alternate drive after the first tape is completely read. After the second tape is read, VPREST will switch back to the primary drive. If an alternate is not specified, only one drive will be used.

BKtape *volser*

requests a specific tape volume for the restore. At the end of the volume, VPREST will ask for the volume serial of the next tape to be used.

CFg *cfgname* (or) **COntig** *cfgname*

a VPARS configuration name. If you specify this option, the database minidisks for the named configuration will be restored. If you do not specify this option (or the VPVDEV option), your default database will be restored. CFG (or CONFIG) and VPVDEV are mutually exclusive.

CYcle *nn*

requests a specific cycle file be used to identify the input tape volumes for this restore.

DEn *density*

the requested tape density.

MDe *xx*

a tape mode set byte. *xx* is the hexadecimal representation of a mode set byte. DEN and MODE are mutually exclusive.

MAp

reads the control information from the beginning of the backup tape, and displays the number and size of minidisks required for the restore. You must have at least as many VPARS minidisks available during the restore as there were during the backup.

OPer *userid*

sends a copy of all tape messages to an operator *userid*.

TApes *n*

requests processing of a specific number of tapes with unspecified volume serial numbers and dataset names. It is the users responsibility to mount the correct backup tapes in the correct order.

VPvdev *vdev*

gives the virtual device number of the base disk of the VPARS database. If you do not specify VPVDEV (or the CFG/CONFIG option), your default database will be restored. VPVDEV and CFG (or CONFIG) are mutually exclusive.

VTape

automatically mounts virtual input tapes to satisfy mount requests, if the restore is being run from virtual tapes. If VTAPE is specified and the tape drive is a real device, or a volume is already mounted on the drive, an error message is issued and the restore terminates.

Usage Notes:

1. VPREST calls an external exec, VSSLTAPE, passing it all tape mount-related messages it issues. You can customize this exec to interface to a tape management system. See the description of the VSSLTAPE exec for more information.
2. You must have a tape drive at virtual device 181. This is checked after the VSSLTAPE exec is passed the Mount message (2301A).
3. If you use the VTAPE option, you should define a virtual tape drive at 181 before calling VPREST. In this case, VPREST will perform the VTAPE mounts; the VSSLTAPE exec does not need to.
4. If neither BKTAPE nor CYCLE is specified, the current cycle in the control file will be used. If control and cycle files do not exist, you will receive an error message.
5. You can look at the cycle file to determine the tape volumes that will be required for a restore.

6. You must have the same number of database minidisks as you had during the VPBKUP. The minidisks do not have to be the same device type, but they must be able to hold all the active data blocks that were backed up.
7. If you do not have enough VPARS minidisks on which to restore the data, the MAP option will be invoked automatically to show you how many disk are required, and the restore will terminate.
8. Your VPARS database must be closed.
9. Existing records that were on your database before the restore may be updated.
10. If you specify an operator userid, a copy of all tape messages will be sent to the designated user. The tape drive address in the messages will be your virtual tape drive, and your userid will be in the message to help the operator relate your virtual drive address to the real address. The operator will receive a copy of all messages related to tape processing, including those requiring a reply, but you must enter the replies.

Responses:

Restore VPARS database? - Enter YES or NO

This verifies the request to restore your VPARS database. If the database should be restored, enter YES, if not, enter NO.

Restoring *vdev*

This response is issued for each minidisk in the VPARS database.

VPARS restore complete, *nnnn* blocks restored

This response is issued after all active database blocks have been restored to the database.

VPUNLD

Use the VPUNLD command to produce a logical backup of all TPF records from your VPARS database to tape.

VPUNLD	<i>vdev altvdev</i> (options options: CFg <i>cfgname</i> OPer <i>userid</i> COnfig <i>cfgname</i> REtpd <i>days</i> DEn <i>density</i> VPvdev <i>vdev</i> MOde <i>xx</i> VTape
--------	--

vdev a tape device number. The default is 181. An asterisk can be used to default the first drive to 181 and specify an alternate. Any valid CMS tape device number can be used (180-187 and 288-28F).

altvdev

an alternate tape device number. If the tape fills up, VPUNLD will switch to the alternate drive. If the second tape fills, VPUNLD will switch back to the primary drive. If an alternate is not specified, only one drive will be used.

CFg *cfgname* (or) **COnfig** *cfgname*

a VPARS configuration name. If you specify a name, records from the database with the requested configuration will be written to tape. If you do not specify CONFIG (or the VPVDEV option), records from your default database will be written.

DEn *density*

a requested tape density.

MOde *xx*

a tape mode set byte. *xx* is the hexadecimal representation of a mode set byte. DEN and MODE are mutually exclusive.

OPer *userid*

sends a copy of tape messages to an operator userid.

VPvdev *vdev*

the starting virtual device number for the input database. You should have a read link to all minidisks in the input database. If you do not specify this option (or the CONFIG option), records from your default database will be written to tape.

VTape

automatically mounts virtual scratch tapes to satisfy mount requests if the unload is being run with virtual tapes. If VTAPE is specified and the tape drive is a real device, or a tape is already mounted on the drive, an error message is issued and the unload terminates.

Usage Notes:

1. The VPUNLD command is similar to the VPBKUP command, but VPUNLD creates a "logical" backup of the TPF records on the VPARS database. When the records are restored with VPLOAD, you do not need to have the same number or size of VPARS minidisks as you had during the unload, as long as all records being loaded will fit on the database.
2. A VPUNLD of a VPARS database will take longer than a VPBKUP, since VPBKUP is able to read the VPARS database a track at a time. VPUNLD reads the database one TPF record at a time.
3. VPUNLD calls an external exec, VSSLTAPE, passing it all tape mount-related messages it issues. You can customize this exec to interface to a tape management system. See the description of the VSSLTAPE exec for more information.
4. You must have a tape drive at virtual device 181. This is checked after the VSSLTAPE exec is passed the Mount message (2301A).
5. If you use the VTAPE option, you should define a virtual tape drive at 181 before calling VPUNLD. In this case, VPUNLD will perform the VTAPE mounts; the VSSLTAPE exec does not need to.
6. Two files are created on your A-disk during an unload: a cycle file that contains the volume serials of the tapes created, and a control file that contains the date, time, and cycle number of the current unload. These files are used by VPLOAD to request input tapes for a load.

The filetype of the cycle file is VPUNLD CYCLnnnn, where nnnn is the cycle number. The filetype of the control file is NCONTROL during the backup. If the unload completes normally, the current control file is renamed to a filetype of OCONTROL and the new control file is rename to a filetype of CONTROL.

7. VPVDEV and CONFIG (or CFG) are mutually exclusive.
8. If neither DEN nor MODE is specified, the density is set to the highest available for the type of tape drive.
9. Use the DEN parameter if you want a density other than 6250 for 9 track tape drives.
10. If you specify an operator userid, a copy of all tape messages will be sent to the designated user. The tape drive address in the messages will be your virtual tape drive, and your userid will be in the message to help the operator relate your virtual drive address to the real address. The operator

will receive a copy of all messages related to tape processing, including those requiring a reply, but you must enter the replies.

Responses:

nnnnn records processed

Issued after every 10,000 records have been unloaded to tape.

VPUTIL

Use the VPUTIL command to delete, list, print, or move TPF records on your VPARS database.

VPUTIL	<p>DElete LIst [[FROM] <i>cc hh rr</i>] [[TO] <i>cc hh rr</i>] (options MOve PRint CHkdir</p> <p>options:</p> <table><tr><td>CONFIg <i>cfgname</i></td><td>ONEmatch</td></tr><tr><td>CONCat <i>cctname</i></td><td>Printer</td></tr><tr><td>Disk</td><td>Terminal</td></tr><tr><td>Hex</td><td>TPFRange <i>vdev vdev</i></td></tr><tr><td>Length <i>n</i></td><td>TPFVdev <i>vdev</i></td></tr><tr><td>Noprint</td><td>Vpvdev <i>vdev</i></td></tr><tr><td>Offset <i>n</i></td><td></td></tr></table>	CONFIg <i>cfgname</i>	ONEmatch	CONCat <i>cctname</i>	Printer	Disk	Terminal	Hex	TPFRange <i>vdev vdev</i>	Length <i>n</i>	TPFVdev <i>vdev</i>	Noprint	Vpvdev <i>vdev</i>	Offset <i>n</i>	
CONFIg <i>cfgname</i>	ONEmatch														
CONCat <i>cctname</i>	Printer														
Disk	Terminal														
Hex	TPFRange <i>vdev vdev</i>														
Length <i>n</i>	TPFVdev <i>vdev</i>														
Noprint	Vpvdev <i>vdev</i>														
Offset <i>n</i>															

DElete

deletes selected TPF records from the VPARS database. A cylinder, head, and record range can be given. Also, a TPFVDEV or TPFRANGE can be used to select the records to be deleted.

LIst lists record numbers of selected TPF records on the VPARS database. A cylinder, head, and record range can be given. Also, a TPFVDEV or TPFRANGE can be used to limit the record numbers listed.

MOve

moves selected TPF records from one database to another. MOVE is generally used to move records from a VPARS database to a set of base TPF disks, which requires write access to the TPF disks. Records can also be moved from any input database (specified with the VPVDEV or CONFIG keywords) to the currently open VPARS database. A cylinder, head, and record range can be given. Also, a TPFVDEV or TPFRANGE can be used to limit the records that are moved.

PRint

prints record numbers and all or part of the content of selected TPF records on the VPARS database. A cylinder, head, and record range can be given. Also, a TPFVDEV or TPFRANGE can be used to limit the records that are printed.

CHkdir

Scan the VPARS database directory structure and verifies that each High Index (II), Index Index (II), Directory Index (DI) and Record Directory (RD) block has the correct header. Although all the options can be specified, only the CONFIG option is taken into account.

FROM *cc hh rr*

identifies the beginning of a cylinder, head, and record range for the TPF records to be selected. The default is 0 0 0. Record or head and record may be omitted. The FROM keyword is optional. The cylinder, head, and record should be entered in hexadecimal.

TO *cc hh rr*

identifies the end of a cylinder, head, and record range for the TPF records to be selected. The default is FFFF FFFF FF. The TO keyword is required if a starting *cc hh rr* is not specified, or if all three positional parameters are not specified on the FROM keyword.

CONFig *cfgname*

a VPARS configuration name. If you use this option, from the specified database will be listed, moved, printed, or deleted. If you do not use this option (or the VPVDEV option), records from your default database will be listed, moved, printed, or deleted.

CONCat *cctname*

a VPARS concatenation name. This option is only valid for the LIST and PRINT functions. When used, all the databases pointed to by the configuration files listed in *concatname* will be searched for the requested records.

Disk writes the listing to disk, named VPUTIL LISTING.

Hex creates the listing file in EBCDIC hex for the PRINT function. The default is to show character and hex.

Length *n*

specifies, in hex, the number of characters to print for each TPF record selected for the PRINT function. The default is to print the entire record. Length is only valid for the Print function.

Offset *n*

specifies, in hex, the offset in the TPF record to start for each TPF record selected for the PRINT function. The default is 0. OFFSET is only valid for the PRINT function.

ONEmatch

Requests that the selected function stops after 1 record has been found. If not specified, the function will continue until all records matching the search criteria are found.

TPFRange *vdev vdev*

a range of virtual device numbers of the TPF disks the records would be

on if VPARS had not intercepted the I/O. If this keyword or the TPFVDEV keyword is not specified, the records for all TPF devices will be selected.

TPFVdev *vdev*

the virtual device number of the TPF disk the records would be on if VPARS had not intercepted the I/O. If this keyword or the TPFRANGE keyword is not specified, the records for all TPF devices will be selected.

Vpvdev *vdev*

the starting virtual device number for the input VPARS database. You should have a read link to all minidisks in the input database. If this parameter or the CONFIG parameter is not specified, your normal VPARS database will be used to print, move, list, or delete records.

Noprint

suppresses the listing file and displays count messages only.

Printer

writes the listing to your virtual printer.

Terminal

writes the listing to your terminal.

Usage Notes:

1. Use VPVDEV or CONFIG to specify a different VPARS database to be used as input. In this way, records can be moved from any input VPARS database to whatever VPARS database you have open. The two databases do not need to have the same number or size of minidisks, as long as all records being moved will fit on the output database.

These options can also be used to move records from any input VPARS database to a read/write set of TPF disks.

Chapter 8. VPARS CMS Execs

VPBXPLTB

Use the VPBXBMAP exec to assemble and execute the VPBXPLTB module and build a TPF pool section directory map.

VPBXPLTB	<i>ctlfn</i>
----------	--------------

ctlfn is the filename for the VPBXPLTB control file. The filetype is VPBXPLTB.

Usage Notes:

1. VPBXPLTB is designed to link and access required macro libraries and assemble and execute the VPBXPLTB module. A sample exec control file TPF24MOD VPBXPLTB and assemble control file VPBXPLTB CNTRL are distributed with VPARS. For a description of the VPARS NOBASE feature, see “Testing Without a TPF Base System” on page 126.

Sample TPF24MOD VPBXPLTB file

```
OUTFILE  TPF24MOD A2
OSMACLIB TPF MACP.RES.V31
OSMACLIB IBM MACP.RES.IBM31
CNTRL    VPBXPLTB
CP       LINK 665151 413 413 RR READ
CMS      ACC 413 X
```

Sample VPBXPLTB CNTRL file

```
TEXT  MACS TPF IBM
TEXT  MACS VSSI32 DMSGPI DMSOM OSMACRO OSPSI
LCP   AUXAVP32 LCP * UPDATES TO VPARS CMS MODULES
```

VSSLTAPE

The VSSLTAPE exec is a tape mount interface for VPBKUP, VPREST, VPUNLD, and VPLOAD. You can customize the supplied VSSLTAPE exec to pass all tape mount messages to a tape management system, or perform any other desired processing.

VSSLTAPE	<i>tape mount related message</i>
----------	-----------------------------------

tape mount related message

The argument passed to the exec will be one of the following CMS messages, documented in the VPARS CMS Messages section of the VPARS User's Guide and Reference. If you also have the VTAPE product installed, and have selected the VTAPE option on the VPARS utility command line, the message that is passed to the exec will be suffixed with \VTAPE. In the following list, *xxx* is VPB for VPBKUP, VPR for VPREST, VPU for VPUNLD, or VPL for VPLOAD.

VSSxxx2301A Mount *vdev* SL [*volser* | SCRATCH]
VSSxxx2302E Tape on *vdev* is not standard labeled
VSSxxx2302R Enter 'M' to Remount or 'T' to Terminate
VSSxxx2303E Tape on *vdev* is *volser1* not *volser2*
VSSxxx2303R Enter 'M' to Remount or 'T' to Terminate
VSSxxx2304E I/O error *n* processing [HDR1 | EOF1] label on *vdev*
VSSxxx2304R Continue? - Enter Y for Yes or N for No
VSSxxx2305E Dsname on tape *vdev dsname*, does not match requested *dsname*
VSSxxx2305R Enter 'M' to Remount, 'U' to Use the tape or 'T' to Terminate
VSSxxx2306I Tape on *vdev* is SL *volser dsname=dsname*
VSSxxx2308W Block count *nn* on tape *vdev* does not match blocks read *mm*
VSSxxx2308R Continue? - Enter Y for Yes or N for No
VSSxxx2309A Keep *vdev* SL *volser dsname=dsname*
VSSxxx2311R Enter 6 character volume serial or 'UNLOAD' for tape on *vdev*
VSSxxx2312I Volume serial entered was *volser*
VSSxxx2312R Correct? - Enter Y for Yes or N for No
VSSxxx2313W Volume *volser*, *dsname=dsname*, on *vdev*, expires *date*
VSSxxx2313R Enter 'M' for Remount or 'U' to Use the tape.

```
VSSxxx2315E Logical error processing [ HDR1 | EOF1 ] label on vdev
VSSxxx2315R Continue? - Enter Y for Yes or N for No
```

Customizing VSSLTAPE

After VPBKUP, VPREST, VPUNLD, and VPLOAD issue any of these messages to the terminal, they will then call the VSSLTAPE exec, passing it the same message. The supplied VSSLTAPE exec will parse the important tokens out of the message (such as the requested virtual tape drive address and the requested tape volume serial), and return to the caller.

You can customize the VSSLTAPE exec to issue a tape mount request to a tape management system, echo the message to another userid, supply responses to the 'R' messages, and so on. The sample VSSLTAPE exec checks for the presence of \VTAPE in the message, and sets a variable if it is found. Your customized exec could then perform different processing based on this condition.

For the messages that require a reply, the customized exec could stack the reply. If it does, that reply will be used. If it does not stack a reply, your terminal will be placed into VM READ and you must type in the reply. Note that the prompt will go to the terminal in either case; if the exec is stacking a response, you may want to also issue a message to the terminal showing what response was stacked. That way, the user won't mistakenly supply a response that doesn't get used.

If you supply a customized VSSLTAPE exec for other users to use, such as on a public disk, you should provide them with instructions showing which messages the customized exec is handling, how it is handling them, and what its responses are, if any, to the messages requiring replies.

Chapter 9. Planning for VPARS

VPARS Database Description

A VPARS database holds modified records for users performing TPF testing. Records that are read or written by the TPF system or application result in virtual machine I/O to a TPF disk. VPARS intercepts this virtual machine I/O. For I/O requests to write a record, VPARS adds the record to the user's VPARS database and updates its record directories.

For I/O requests to read a record, VPARS searches its record directories. If the TPF record is found on the VPARS database, the record is returned to the application. If the record is not found on the VPARS database, the I/O request is passed through to VM, resulting in the record being read from the TPF base disk. This database shadowing activity is invisible to the system or application in the virtual machine.

A VPARS database can contain up to 256 minidisks of any size. VPARS supports all DASD devices supported by VM. For best performance, all VPARS minidisks for a user should be the same size, and each minidisk should be on a different real device.

Database Device Numbers

The virtual device number of the first VPARS database disk and the maximum number of disks to be included in the database are assigned in VPARS configuration files. The device number of the first minidisk must end in zero, and should not be in the same device number range as any device used by TPF or CMS. For 370 mode guest machines, the starting address can be any number up to 1FF0. For XA and ESA mode guests, the starting device number can be any number up to FFF0. Each TPF guest can have up to 256 VPARS database minidisks to hold modified records. However, if the starting device number is FFF0, the database can only contain 16 minidisks. If you want to use 256 minidisks in the database, the largest starting device number is FF00. Additional minidisks in the database must be numbered sequentially. For example, a database whose starting device number is 1F00 and contains 64 minidisks would be numbered 1F00, 1F01, 1F02, ... 1F3F.

When a user opens a database, VPARS searches the user's minidisks. The number of minidisks VPARS will initially use is determined when an empty device number is found in the database device range. Formatted VPARS minidisks can be added to the database after it is opened, to provide additional capacity.

You can define the maximum number of database minidisks to be less than 256. Unless a user requires a very large database, it is common to define databases to contain 16 minidisks. As an example, a 16-minidisk database might contain minidisks from 1200 to 120F, or 1FF0 to 1FFF.

When VPARS opens a database for the first time, it assigns a unique key to all disks in that database, and the disks are numbered in sequence. VPARS verifies the key and sequence on subsequent opens. If the disks are not in sequence or the key does not match, VPARS will not open the database.

Database Minidisk Allocation

These rules should be followed when you allocate VPARS database minidisks:

- Each user's VPARS database should be split into equal sized minidisks on as many different real devices as possible. A user should not have more than one database minidisk on the same real device, unless it is a solid-state disk drive.
- VPARS stores data in 4096-byte data blocks. Each new data block is written to the minidisk with the least number of blocks allocated. If you are using solid state devices, you may want to create several database minidisks on the solid state device and one large minidisk to a non solid state device. This will cause more blocks to be allocated to the solid state device. You should spread the allocation of the VPARS minidisks across all available solid state paths.
- Highly active database minidisks should be allocated in the center of real (movable-head) devices.
- A real disk drive containing VPARS minidisks should be dedicated to VPARS use. VPARS issues long chains of CCWs to its disks, like system paging. These long CCW chains can severely impact other functions attempting to use the same real device. Therefore, real disks containing VPARS database minidisks should not hold paging, spooling, or frequently used CMS user mindisks. Also, other functions using the same real devices as VPARS can impact the user's performance under VPARS.
- A real device containing VPARS database minidisks should not be shared by more than one processor.
- On a system with heavy paging loads, disks containing VPARS database minidisks should not be on the same controller or channel path as paging devices.

Database Minidisk Sizes

The disk space required for a database will vary depending on the type of device used and the number of TPF records modified by a user. The database should be large enough to hold all modified records for most tests.

Figure 1 on page 125 shows the maximum capacity of a cylinder for 4096, 1055, and 381 byte records. The actual number of records written on a cylinder will vary because 381 and 1055 byte records are intermixed in 4096-byte VPARS data blocks.

Figure 2 on page 125 shows suggested starting sizes for VPARS minidisks, and the record capacities of these minidisks.

If a user requires more capacity than is provided by one minidisk, additional minidisks can be added to the user's VM directory entry to provide the required capacity. VPARS performance is improved if the database consists of more than one minidisk.

Device	3340	3350	3375	3380	3390
381	380	1200	960	1500	1800
1055	114	360	288	450	540
4096	38	120	96	150	180

Figure 1. Maximum records per cylinder.

Device Cyls	3340 250	3350 50	3375 60	3380 40	3390 40
381	95,000	60,000	57,600	60,000	72,000
1055	28,500	18,000	17,280	18,000	21,600
4096	9,500	6,000	5,760	6,000	7,200

Figure 2. Recommended VPARS minidisk sizes and capacities.

Multi-Level Databases

Multi-level (or "concatenated") VPARS databases allow several databases to be searched for a TPF record. The primary database is normally a read-write database, and the others are always read-only databases. Several users can share each read-only database. This may reduce the number of records required on each VPARS user's read/write database, since a shortload or longload (or any other data) can be written to a database which is then shared read-only among several VPARS users. The primary database can also be a read-only database.

Loosely-Coupled Testing

VPARS also supports testing of loosely-coupled TPF systems. All TPF virtual machines participating in a loosely coupled test write to the same VPARS database. Each virtual machine must have write links to all disks in the

database. VPARS provides only the multi-write capability; record locking is the responsibility of the application. IBM's Limited Lock Facility PRPQ or the Multi-Path Lock Facility (MPLF) can be used to provide record locking.

The TPF Base System

The device numbers of the TPF base disks are defined in VPARS configuration files. VPARS users should link to these disks read/only. When the user opens a VPARS database, VPARS flags these disks to have their I/O intercepted. The disks are internally changed to read/write to allow CCW translation to take place. VPARS will intercept and process all I/O requests for these disks.

The virtual device numbers of the TPF disks are used in the VPARS record directory to locate TPF records. If you change a TPF virtual device number, VPARS will not be able to locate the data for that volume when I/O is performed to the new device number. If you link a different TPF volume at the old device number, the records for the old volume will be retrieved when I/O is done to the new volume.

TPF Record Sizes

VPARS can use TPF format tables to verify the size of each record to be added to the VPARS database, if desired. These tables are defined in assembler files, or created by running the VPBLDFMT command against a formatted TPF disk, and moved to the VSSI parm disk. The tables contain all of the information required by VPARS to know the exact format of the TPF system disks.

The first time a record is written, VPARS uses the format table for the TPF disk to determine the correct length for the record. The next time the record is read or written, VPARS uses the length of the record on the database for length checking.

If the length requested in a read or write CCW does not match the length specified in the format table (or the length of the previously written record), the CCW will be rejected with an incorrect length, unless the suppress incorrect length flag (SILI) is set in the CCW. Record length checking does not apply to records on track zero.

If format tables are not used, VPARS does not perform record length checking. In this case, wrong length records which would have been rejected in a native TPF system will not be rejected by VPARS.

Testing Without a TPF Base System

The purpose of the VPARS NOBASE feature is to provide a TPF online environment that requires less disk space than a half system test system. This is accomplished by restoring only the active online records to a VPARS database. Short term and unused long term pool records are not restored. This VPARS

database can then be shared by several users with multilevel VPARS databases, and no TPF base system is required.

As an example of potential space savings, a VPARS NOBASE database was built from a 592 module 3380 TPF system. The resulting VPARS database used the equivalent space of 267 3380 disks. It actually resided on 128 3390 model 3 disks using 2000 cylinders on each 3390. The database was 77 percent full.

Several CMS utilities have been written to build a NOBASE VPARS database (VPBXREST, VPBXPLTB, and VPBXBMAP), without restoring short term and unused long term pool records. See the description of the VPBXREST for more information. The space savings can be significant, but setting up a NOBASE system can be complex.

The VPARS user must still have a link to a disk of the correct device type (3380, 3390, etc.) at each device number to which TPF may issue I/O requests. If not, VM's CCW translation will reject every I/O request before VPARS can intercept them. The disks do not have to be in TPF format, and VPARS will not issue any I/O requests against the disks linked at the base system device numbers, except to retrieve IPL text. A single real disk can be linked multiple times to provide a link at all required TPF device numbers.

All relevant TPF records should be loaded to the VPARS database. When running with a NOBASE configuration, VPARS will respond to any read of a record that is not found on the database by returning a zero record (consisting of binary zeros for the specified length). A format table is required when using a NOBASE database.

Backups and Restores

VPARS provides two forms of backup and restore functions. The first set of functions, VPBKUP and VPREST, saves (to tape) and restores all database blocks which contain data. VPBKUP produces a physical backup of a VPARS database. VPREST restores these records to the same relative minidisk from which they were backed up. Therefore, VPREST requires as many minidisks in the database as there were during the backup. The minidisks do not have to be the same disk type, but each one must be large enough to hold the data that was backed up from the same disk by VPBKUP.

Two additional functions, VPUNLD and VPLOAD, are provided to unload and load TPF records from a VPARS database to tape. VPUNLD produces a logical backup of a VPARS database by reading the TPF records from the database and writing them to tape. The structure of the VPARS database (the number and size of minidisks) is not part of the logical backup. TPF records from a tape created by VPUNLD can be loaded to any configuration of VPARS database, as long as it is large enough to hold the records.

Note that a tape created by VPUNLD can also be loaded to a base TPF system, if you have read/write access to the TPF system disks. VPLOAD works by reading the TPF records from the unload tape, and writing them to their original location on the TPF base system. If the VPARS database is open, VPARS will

intercept the writes. If the database is closed, VPLOAD will write to the TPF base disks.

Chapter 10. Customization Overview

After VPARS is installed in your system as described in the Product Installation Guide, you must create several configuration files, and add or update entries in the CP directory to customize your VPARS system. Configuration files are standard editable fixed 80 byte record text files, (with the exception of the VPTPFMT file which is a binary text file), and are stored on the VSSI parm disk(s).

VPARS uses five types of configuration files:

- System Defaults
- Configuration Definitions
- Concatenation Definitions
- TPF Device Number Tables
- TPF Device Format Tables

The VPQUERY command can be used to list the names of VPARS configuration files. It can also be used to display information in online configuration files.

The VPARS configuration file names are:

Filename	Filetype	Where located
VPSYSTEM	DEFAULTS	MUST be on the primary VSSI parm disk
<i>cfgname</i>	VPCONFIG	The other files
<i>cctname</i>	VPCONCAT	can be on
<i>devname</i>	VPTPFDEV	the primary or
<i>fmtname</i>	VPTPFMT	secondary parm disk

The primary VSSI parm disk is the disk defined by the initialization statement VSI_Disk. Secondary disks can be assigned to userids or CP classes by coding the following keywords in the VPSYSTEM DEFAULTS file:

- PARM_USerid or
- PARM_Class

Note: If a user has a specific parm disk assigned, all the files required must be on that disk. VPARS will not search the primary parm disk for that user.

In order to properly set up the configuration files, you should understand how the files are used. The following list describes the logic of the VPOPEN command:

1. If the user specifies a VPARS configuration name on the VPOPEN command, VPARS will use that configuration.
2. If not, VPARS will check the system defaults table (defined in VPSYSTEM DEFAULTS) to see if the userid (or userid prefix) appears in the USERID_config list. If it does, then VPARS uses the corresponding configuration. If the userid or prefix does not appear, VPARS uses the default configuration name from VPSYSTEM.

3. VPARS reads the configuration file to obtain the device number of the first disk and the maximum number of disks in the VPARS database.
4. If NOCONFIG is specified, there must be a system default concatenation, or the user must specify a concatenation with VPOPEN.
5. If a TPF device table name was not entered on the VPOPEN command, VPARS uses the device table name in the configuration definition. If there is no device table name in the selected configuration, the default device table name from VPSYSTEM is used.
6. If a concatenation name is entered on the VPOPEN command, or if there is a default concatenation for the system or for this userid prefix, VPARS reads the concatenation definition file, then reads the configuration file for each configuration named in the concatenation.
7. The link mode is checked for each VPARS database minidisk. If RDONLY (read/only) was not requested, the database disks for the main configuration must be linked read/write. If a concatenation was requested, the database disks for each configuration named in the concatenation must also be linked (read/only or read/write).
8. An eight digit key is built for each database being opened. The key is based on the real location of the first minidisk in each VPARS database. The first four digits are the device number (address) of the real disk containing the first minidisk. The last four digits are the starting cylinder number of the minidisk on the real disk.
9. The keys of the databases being opened are checked to ensure that they do not conflict with any other open database. If a database is open by other users for input, it cannot be opened by this user for output (unless this user is authorized by having a command class that is listed on the OPEN_RO_FOR_Output keyword of the configuration file for this database). If a database is open by another user for output, it cannot be opened by this user for output.
10. For loosely coupled testing, each user issuing the VPOPEN command must specify the COUPLED parameter. In this case, multiple users are allowed to open the same database for output. Each user must have a write link to the database disks.
11. If the read/write tests are met, VPARS opens the requested databases.
12. VPARS uses the selected TPF device table to flag each disk in the base TPF system to have its I/O intercepted. VPARS also reads the TPF format table that corresponds to each device range.

You should provide users with links to the minidisks that make up their default VPARS database, either in their directory entry or in a setup-type exec.

Chapter 11. Creating Configuration Definitions.

The VPARS System Defaults File

The system defaults file defines systemwide VPARS defaults.

A VPSYSTEM DEFAULTS file must be loaded before any other configuration files can be loaded. Also, you may want to issue the VPINIT command whenever you replace the VPSYSTEM DEFAULTS file. See VPINIT for more information.

```
CONFIG_TYPE *SYS
CFG_Default cfgname *
CCT_Default concatname *
DEV_Default devname *
SETMax_users ccc B
MAXUSers nnn 150
PCTFull nn 90
POol_md_userid userid VMDPOOL
SETOthers_auth ccc B

USER_config userprefix cfgname cctname
USER_config userprefix cfgname cctname
..

PARM_USerid userprefix userid vdev
PARM_USerid userprefix userid vdev
..

PARM_CClass ccc userid vdev
PARM_CClass ccc userid vdev
..
```

CONFIG_TYPE *SYS.

Defines the type of the file (*SYS for VPSYSTEM). This MUST be the first record in the file.

CFG_Default *cfgname*

specifies the name of the default VPARS configuration. This configuration is used if the user does not specify a configuration name (or specifies NOCONFIG) on the VPOPEN command, and the userid prefix does not appear on any USER_config keywords. Specify 'CFG_Default *' or omit the keyword entirely if you do not want to define a default configuration. The default is *.

CCT_Default *cctname*

specifies the name of the default VPARS concatenation. This concatenation is used if the user does not specify a concatenation name (or specifies NOCONCAT) on the VPOPEN command, and the userid prefix does not appear on any USER_config keywords. Specify 'CCT_Default *' or omit the keyword entirely if you do not want to define a default concatenation. The default is *.

DEV_Default *devname*

specifies the name of the default TPF device table. This table is used if the selected configuration file does not contain a device table name.

Specify 'DEV_Default *' or omit the keyword entirely if you do not want to define a default TPF device table. The default is *.

SETMax_users *ccc*

one or more letters that represent the command classes of users who are allowed to use the VPSET MAXUSERS command. The default is B.

MAXUsers *nnn*

specifies the maximum number of users allowed to have open VPARS databases. The limit can be displayed online with the VPQUERY MAXUSERS command, and changed with VPSET MAXUSERS. The default is 150.

PCTFull *nn*

specifies the percent full of a VPARS database when the first warning message is issued. A message is also issued for each percentage used above this value. The default is 90.

POol_md_userid *userid*

specifies the userid of a directory entry that contains MDISK statements for a pool of formatted VPARS minidisks. The default is VPMDPOOL.

SETOthers_auth *ccc*

one or more letters that represent the command classes of users who are allowed to use some VPSET commands on behalf of other users. The default is B.

USER_config *userprefix cfgname cctname*

Links a userid prefix (or userid), with default configuration and concatenation names. For example, if the prefix is VPTPF, any user whose userid starts with VPTPF will use the specified configuration and concatenation as a default. Note that a user can always specify any configuration or concatenation name during VPOPEN. If a name is not given during VPOPEN, the userid prefix list is searched to find a configuration and concatenation name. If the userid or prefix is not found in this list, the system default configuration and concatenation names are used.

Either the configuration or concatenation name (or both) can be omitted. For example, you may want to specify a default configuration name but no default concatenation name, for one or more user groups. If the userid prefix is found during VPOPEN processing, but the configuration or concatenation name is omitted, VPARS will check the system defaults to see if there is a configuration or concatenation named there.

You need one USER_config keyword per *userprefix cfgname cctname* set. If you do not need different default names for different groups of users, just omit the USER_config keyword(s).

PARM_USerid *userprefix userid vdev*

Links a userid prefix (or userid) with its own VSSI parm disk, as defined by the owning *userid* and target *vdev*

Note: The mini-disk must be CP accessed before VPARS can access the files. Also, note that all the required files (configuration, concatenation, device table and format table) must be on the specified disk. Vpars will not search the master parm disk (the one defined at initialization time with the keyword VSI_DISK) for missing files.

PARM_Class *ccc userid vdev*

Links CP privilege classes with a given VSSI parm disk, as defined by the owning *userid* and target *vdev*

Note: The mini-disk must be CP accessed before VPARS can access the files. Also, note that all the required files (configuration, concatenation, device table and format table) must be on the specified disk. Vpars will not search the master parm disk (the one defined at initialization time with the keyword VSI_DISK) for missing files.

VPARS Configuration Definitions

The configuration definition files define the individual configurations. A user can specify a configuration name on the VPOPEN command, or the user's default configuration can be used for the open. The configuration definitions also define the parts of a concatenation, since a concatenation definition is a series of configuration names. The system-wide default configuration, and the defaults for users with specific userid prefixes, are specified in the VPSYSTEM DEFAULTS file.

The filename of each file that defines a configuration will be the name of the configuration.

CONFIG_TYPE	*CFG	<u>N/A</u>
ACTIVITY_INTERVAL	<i>seconds</i>	<u>15</u>
BASEvpars	<i>vdev</i>	<u>1FF0</u>
VPMD_max	<i>nn</i>	<u>16</u>
BASE_system	YES or NO	<u>YES</u>
CLEAR	<i>option</i>	<u>NORMAL</u>
DBQ_LOOKUP	<i>option</i>	<u>SEARCH</u>
ECHO	YES or NO	<u>NO</u>
LCP_unlike	YES or NO	<u>NO</u>
LOCK_STAT_interval	<i>seconds</i>	<u>15</u>
VPPOOL_max	<i>nnn</i>	<u>256</u>
POOL_DISK_LIMIT_size	<i>t s m</i>	
VPUSER_max	<i>nn</i>	<u>32767</u>
MAXONQ	<i>nn</i>	<u>1000</u>
MINONQ	<i>nn</i>	<u>50</u>
MINONQ_USER	<i>nn</i>	<u>25</u>
NUMBER_OVER_MINONQ	<i>nn</i>	<u>25</u>
OPEN_RO_FOR_Output	<i>ccc</i>	<u>P</u>
PERCENT_OVER_MAXONQ	<i>nn</i>	<u>75</u>
PERCENT_OVER_MINONQ	<i>nn</i>	<u>25</u>
POOLSIZE_default	<i>nn</i>	<u>0</u>
RDQ_lookup	<i>option</i>	<u>SEARCH</u>
REQ_ONline	YES or NO	<u>NO</u>
SECONDS_OVER_MINONQ	<i>seconds</i>	<u>5</u>
VPSTAT_default	<i>minutes</i>	<u>15</u>
TPFDEV_Table	<i>devname</i>	
TRACKZero	<i>option</i>	<u>KEYED</u>

CONFIG_TYPE *CFG.

Defines the type of the file (*CFG for Configuration). This MUST be the first record in the file.

ACTIVITY_INTERVAL *seconds*

specifies the interval for collecting activity data. This parameter smooths the per second values for activity information. The VPQUERY ACTIVITY command can display activity information by second, interval or session. The default is 15 seconds.

BASE_system YES or NO

specifies whether there is a base TPF system for this VPARS configuration. This is normally YES. Specify NO for a NOBASE database as described in “Testing Without a TPF Base System” on page 126. The default is YES.

BASEvpars_md *vdev*

specifies the device number of the first VPARS minidisk in a database when this configuration is used. This device number must end in zero, as described in “Database Device Numbers” on page 123. The default is 1FF0.

CLEAR NORMAL or RESTRICTED

specifies whether the VPCLEAR, VPOPEN CLEAR, and VPIPL CLEAR commands are restricted in this database. See the VPCLEAR command description for a full explanation of the RESTRICTED option. The default NORMAL.

DBQ_lookup SEARCH or GAL

specifies the type of search to be done on the Data Block Queue when a requested record is found on the VPARS database. Search should be used for databases with a small MINONQ value (100 or less). GAL is the Generalized Associative Lookup facility of VM CP. The default is SEARCH.

ECHO YES or NO

specifies whether records that are read from the base TPF system should be rewritten to the VPARS database. This makes it easier to build a database that can later be used as a NOBASE database, since running an application with "read echo" ensures that all records that are required for that test are written to the database. The default is NO.

LCP_unlike YES or NO

specifies whether or not a user who is trying to "join" a loosely-coupled multi-level database is required to have the same set of read/only database levels as the other users who are already coupled to the database. If there is a mismatch, incorrect data records could be returned to the TPF application, causing application errors. This parameter controls whether or not the read/only levels are checked for consistency with the read/only levels of the other users who are coupled to the database. The default is NO

LOCK_Stat_interval *seconds*

specifies the interval for collecting activity lock status data. Lock status data is performance information on the Record Directory and Data Block Queue locks. This parameter smooths the per second values for lock status information. The VPQUERY LOCKQUEUE command can display activity information by second, interval or session. The default is 15 seconds.

VPPool_max *nnn*

the maximum number of pool disks that any user will be allowed to link for this configuration using the VPLINK command. The default is 256.

POOL_DISK_LIMIT_size *t s m*

the maximum number of pool disks by type (3380, 3390, etc.) and minidisk size in cylinders that any user will be allowed to link for this configuration using the VPLINK command. Specify one set of three parameters per POOL_DISK... keyword. The first item is the device type. The second item is the size in cylinders. The third item is the maximum number of disks of that type and size that any user will be allowed to link. You do not have to code this keyword. The default is no list.

VPUSer_max *nn*

the maximum number of users that are allowed to open a database that was opened by this VPARS configuration. Specify a number between zero and 32767. If zero is specified, the data base cannot be opened. The default is 32767.

MAXONQ *nn*

is a tuning parameter that specifies the maximum number of VPARS buffers on the Record Directory and Data Block queues. This number is used as a separate limit for each queue. We recommend that you use the default value until you evaluate your buffer usage with the VPQUERY BUFFERS command. The default is 1000 buffers.

VPMD_max *nn*

specifies the maximum number of VPARS minidisks when this configuration is used. BASEvpars_md and VPMD_max define the virtual device number (address) range that will be used when any VPARS database is opened using this configuration. Any gap in the range ends the search for disks to include in the database. The value specified must be a number from 1 to 256. The default is 16.

MINONQ *nn*

is a tuning parameter that specifies the lower limit of VPARS buffers on the Record Directory and Data Block queues. This number is used as a separate limit for each queue. We recommend that you use the default value until you evaluate your buffer usage with the VPQUERY BUFFERS command. The default is 50 buffers.

MINONQ_USER *nn*

is a tuning parameter that specifies the number of buffers to be added to MINONQ for each user that joins a loosely coupled or concatenated VPARS database. The default is 25 buffer.

OPEN_RO_FOR_Output *ccc*

lists the command classes of users who are allowed to open a database for output that is currently open for input by other users. If several users have a database open for input, one user will be allowed to open that database for output, if that user has one of the command classes in this list. The default is P.

PERCENT_OVER_MAXONQ *nn*

is a tuning parameter that specifies the percentage of MAXONQ buffers that can be in use before a buffer release is done. The default is 75 percent.

PERCENT_OVER_MINONQ *nn*

is a tuning parameter that specifies the percentage over MINONQ buffers that can be in use before a buffer release is done. The default is 25 percent.

POOLSize_default *nn*

the default pool minidisk size in cylinders for this configuration. This value is used by the VPLINK command to link a minidisk of a specific size from the directory entry for the VPARS pool minidisk definitions. This userid is specified on the POol_md_userid keyword in VPSYSTEM. A value of 0 (zero) implies any size. The default is zero.

RDQ_LOOKUP_SEARCH *or GAL*

specifies the type of search to be done on the Record Directory Queue when a requested record is found on the VPARS database. Search should be used for databases with a small MINONQ value (100 or less). GAL is the Generalized Associative Lookup facility of VM CP. The default is SEARCH.

REQ_ONline YES *or NO*

specifies whether a database opened with this configuration should call a user interface module (HCPVO1) to request records that are not found in the VPARS database. Normally, records that are not found are read from the base system disks. HCPVO1 is provided to allow a programmer to include code that will request a record from an online system (or from any other source). The default is NO.

SECONDS_OVER_MINONQ *nn*

is a tuning parameter that specifies the time that MINONQ can be exceeded before a buffer release is done. The default is 5 seconds.

VPStat_default *minutes*

the number of minutes between automatic VPQUERY status displays while a database is open. Specify a number between zero and 32767. If zero is specified, no automatic status displays will be done. The default is 15 minutes.

TPFDEV_Table *devname*

the name of the TPF device table to be used for this configuration. If the system default TPF device table name defined in VPSYSTEM is correct for this configuration, you do not need to respecify the name here. You may omit this parameter or specify 'TPFDEVTBL *!.

TRACKZero KEYED *or NOTKEYED*

specifies whether records on track zero of the disks being intercepted by VPARS are keyed or not keyed. KEYED should be specified for TPF or MVS disks. NOTKEYED should be specified for CMS disks. The default is KEYED.

VPARS Concatenation Definitions

A multi-level VPARS database consists of a series of individual databases, referred to by their configuration names. A VPARS concatenation definition file specifies the names of the configurations that make up a concatenation. The filename of each file that defines a concatenation will be the name of the concatenation. The name is used to specify the concatenation during VPOPEN. Each of the databases in the concatenation will be opened, and you must have VPARS database disks in each device range (as defined in each configuration definition).

With a multi-level database, each time the TPF test system issues a read I/O request, VPARS searches each of the databases for the requested record (in the order they are listed in the concatenation definition). If the record is not found in any of the databases, it is read from the TPF base system. When a write I/O request is issued, the record is written to the primary read/write database.

Note that when a user opens a concatenated database, a primary read/write database will normally be opened in addition to the read/only levels defined in the concatenation. The read/write database will be the user's default (or explicitly specified) configuration name. Therefore, you do not need to specify the primary database configuration name in the concatenation definition; it should only specify the read/only levels.

The user also has the option of opening a multi-level database and omitting the primary read/write level, or opening the primary level read/only along with the databases defined in the concatenation. See the VPOPEN command usage notes for more information.

```
CONFIG_TYPE  *CCT
CONCAT       cfgname cfgname cfgname ....
```

CONFIG_TYPE *CCT.

Defines the type of the file (*CCT for Concatenation). This MUST be the first record in the file.

cfgname

a list of one to sixteen configuration names that makes up this concatenation.

TPF Device Table Definitions

The TPF device table definitions identify the device numbers of TPF disks that will have their I/O intercepted by VPARS, and the name of the format table to be used for the disks in the device number range. The filename of each file that defines a TPF device table will be the name of the device table. The name is used to assign a default device table in VPSYSTEM DEFAULTS, and to specify the device table on the VOPEN command or in a configuration definition.

```
CONFIG_TYPE  *TPF
DEVtbl      vdev-vdev fmttblname
```

CONFIG_TYPE *TPF.

Defines the type of the file (*TPF for TPF device). This **MUST** be the first record in the file.

vdev-vdev

the device number range of a set of TPF disks that share a format table. All read/only disks in this range will have their I/O intercepted by VPARS.

Note: You need one DEVtbl keyword per device range.

fmttblname

the name of a VPARS format table. Different device ranges can reference the same format table.

TPF Format Table Definitions

The TPF Format Table definition files define the device type and disk format for ranges of device numbers in a base TPF system. The filename of each assemble file that defines a TPF format table will be the name of the format table. The name is used to assign a format table to a device range in a TPF device table.

You can define and assemble a format table definition file, or you can use the VPBLDFMT CMS command to create a format table definition text file from a formatted TPF disk.

Note: The TPF format table is the only VPARS configuration file which needs to be assembled (if not created by VPBLDFMT). The resulting TEXT file needs to be copied to the VSSI parm disk with a filetype of VPTPFMT for VPARS to use it

```
VPFMTMAC DEVTYPE=devicetype,  
            (startcyl,starttrk,endcyl,endtrk,recsize,recincr,rbits),  
            (startcyl,starttrk,endcyl,endtrk,recsize,recincr,rbits),  
            (startcyl,starttrk,endcyl,endtrk,recsize,recincr,rbits)
```

DEVTYPE=*devicetype*

identifies the type of disk device that this format table defines. Device types are shown in Figure 3 on page 141.

startcyl,starttrk

the starting cylinder and track number for a range of tracks that contains one record type.

endcyl,endtrk

the ending cylinder and track number for a range of tracks that contains one record type.

recsize

the size of the records in this range (381, 1055, or 4096).

recincr

the record number increment for records on this track. In most systems, this is 8 for 1055 byte records, 4 for 381 byte records, and 1 for 4096 byte records. For 3390 disks, an increment of 1 is normally used for all record sizes.

rbits the value of the low-order two bits of the record numbers for TPF records in this range.

Values for the cylinder and track ranges can be obtained from the TPF format deck that was used to format your TPF system disks.

You can code a TPFTBL with no cylinder ranges, although the device type is still required. If you use this option, VPARS will *not* do record length checking when records are added to the database. Default record increments of 8 for 1055 byte records, 4 for 381 byte records, and 1 for 4096 byte records will be used to increment the record number for CCW chains that contain multiple read/write CCWs. Record increments of 1 are used for all record sizes for 3390s.

#

Note: The previous statement *DOES NOT* apply if ESS dasds (aka:Shark) are used. The support code for ESS dasd *REQUIRES* a fully populated format table.

Devtype	Device	Max size
2305-1	2305	48 cylinders
2305-2	2305	96 cylinders
3330	3330	404 cylinders
3330-2	3330	808 cylinders
3340-1	3340	348 cylinders
3340-2	3340	696 cylinders
3340-3	3340	1000 cylinders
3344	3340	696 cylinders
3350	3350	555 cylinders
3375	3375	959 cylinders
3380	3380	885 cylinders
3380A	3380	885 cylinders
3380D	3380	885 cylinders
3380J	3380	885 cylinders
3380E	3380	1770 cylinders
3380K	3380	2655 cylinders
3390	3390	1113 cylinders
3390-1	3390	1113 cylinders
3390-2	3390	2226 cylinders
3390-3	3390	3339 cylinders
3390-9	3390	10017 cylinders
9345-1	9345	1440 cylinders
9345-2	9345	2156 cylinders

Figure 3. Valid device types in the VPFMTMAC macro.

Chapter 12. Post-Installation Customization

After you have installed VPARS as described in the product installation guide, you will create several customization and configuration definition files.

1. Customize the VPSYSTEM SAMPLE file, or convert the VPSYSTEM definition file from your previous release of VPARS.
2. Move your customized VPSYSTEM DEFAULTS file to the primary VSSI parm disk. (The disk defined by the VSI_Disk initialization statement)
3. For the remaining files, keep in mind that they may not necessarily reside on the primary VSSI parm disk. It will depend whether or not you coded the PARM_USerid or/and PARM_CClass statements in the VPCONFIG DEFAULTS file.
4. Create the configuration definitions that will be required for your system. The sample configuration definition is called CONFIG1 CONFIG.
5. Move the configuration definitions to the appropriate VSSI parm disk.
6. If you will be using multi-level VPARS databases, create the concatenation definitions. The sample concatenation definition is called CONCAT1 CONCAT. You can create the concatenation definitions at a later time if desired.
7. Move the concatenation definitions to the appropriate VSSI parm disk.
8. Create the TPF device table definitions. The sample device table definition is called TPFDEV1 VPTPFDEV.
9. Move the device table definitions to the appropriate VSSI parm disk.
10. If you will be using format tables, create the TPF format table definitions. The sample format table definition is called FMT3380A ASSEMBLE.
11. Assemble the format table definition (using VSSASM) and move the resulting text deck to the appropriate VSSI parm disk.
12. If you are installing VPARS for the first time, you should create userids to be used for TPF testing under VPARS. Provide these users with minidisks to be used as VPARS databases. Sample directory entries are in a file called VPSAMPLE DIRECT32.
13. Each user should format their database minidisks with the VPFMT command.
14. If you have a VPARSVMB buffer virtual machine userid in your directory from a previous release of VPARS, you can remove it. VPARS now uses system virtual storage to buffer database I/O.
15. If you have a VSSINIT exec from a previous release of VPARS, you should remove it from your system startup procedure. It is no longer necessary to

autolog a buffer userid for VPARS. (If you also have the VTAPE product, it is no longer necessary to autolog a buffer userid for VTAPE either.)

You can now open a VPARS database and follow your normal procedures to IPL a TPF system.

Chapter 13. VPARS Modifications to VM

VPARS uses one of the 'VMDUSERx' field of the VMDBK (Default is VMDUSER8. see Installation guide and VPOPTNS MACRO) to maintain a pointer to a control block containing information about a user's open databases. This field should not be used by any other package installed in your system.

VPARS updates several CP modules to interface to VPARS. All source statements added to VM are flagged with '\$VPnnnn' in columns 65-71 of the source code, where *nnnn* is an update number. These modifications are described in the VPARS VMMODS listing file on the VSSI installation disk.

Chapter 14. PTV Support

Description

The PTV mode of VPARS is an optional feature of VPARS, and can be used to eliminate the requirement for a Data Base Restore (DBR) tape during PTV processing. PTV mode is turned on or off with the VPSET command, and can also be turned off with VPCLEAR.

When PTV mode is active, all keypoint and PTV records are maintained in a special record directory on the VPARS database. These TPF records are not cleared from the database with the normal VPCLEAR and VPIPL CLEAR commands. This allows keypoint and PTV records to be maintained across database clears. When PTV mode is turned off, the keypoint records and the database are cleared.

Method of Operation

The modification to PTV processing is designed to allow PTV to run with or without a DBR tape. The DBR tape can be eliminated only if PTV is running in a virtual machine under VPARS in PTV mode. Minor modifications are also required to the TPF PTV programs.

Installation

To use PTV mode for the elimination of the DBR tape, a file resident TPF program must be allocated and loaded to your TPF system. Three PTV programs must be modified to enter the added program during PTV processing.

The added PTV program and the updates for the modified programs are distributed on file 7 of the VPARS distribution tape. The added program is distributed with a file name of 'PTV9VP ASSEMBLE'.

The field EBW092 of the PTV ECB is used to communicate between the modified and added PTV programs.

PTV Program Modification

The update files for the PTV programs are distributed in CMS update format.

1. PTVB enters the added program between test units to determine if programs loaded by the prior test unit must be restored. Program restore is not required if the database was cleared after the prior test unit.

The update for PTVB is on file 7 with a file name of 'PTVBVP UPDTVPRS'. The update deck is sequenced for ACP 9.2.1.

Source statements are added to PTVB just prior to the ENTRC to PTVM for program restore. The ENTRC for PTVM is in the routine at tag PTVRES just prior to the tag PTVCLR.

2. PTVX enters the added program to determine if the DBR tape should be mounted and the DBR macro table entries initialized.

The update for PTVX is on file 7 with a file name of 'PTVXVP UPDTVPRS'. The update deck is sequenced for ACP 9.2.1.

Source statements are added to PTVX immediately after the register save at entry to PTVX.

3. APTV enters the added program prior to IPL simulation. If PTV mode is not active or the next test unit requests the no restore option, control returns to APTV for normal IPL simulation; otherwise, a diagnose instruction is issued to IPL the system and clear the database. The checkpoint clear option is used with the IPL to allow PTV to be run with a program base other than the one on the base test system.

The update for APTV is on file 7 with a file name of 'APTVVP UPDTVPRS'. The update deck is sequenced for ACP 9.2.1.

Source statements are added to APTV just prior to the CINFC to CM1IPL after tag BRSALD in the IPL simulation routine.

Chapter 15. Concatenation Tutorial

The purpose of concatenated VPARS databases is to provide different test system images to different test system users, using a common TPF base system. Also, records that need to be shared by several users, one department, or all users can be loaded to one database rather than being loaded to each user's read/write database. Eliminating this duplicate loading of records can save considerable disk space.

This tutorial on using concatenated VPARS databases will assume that a TPF programming department has two programming groups, each group has two teams, and each team has two users. Each user will have a read/write VPARS database and three levels of concatenation. The concatenation order of the databases will be the user's R/W database, then the Team database, then Group, and then Department.

We will define a primary and an alternate set of minidisks for each VPARS database. This will allow a maintenance group to build a new VPARS database for one concatenation level while the other copy of that database is being used for testing. CMS execs will be used to control which set of minidisks are linked for each database in a user's concatenation.

Database Structure

As you go through this tutorial you may find it helpful to reference “Concatenation Example 1” on page 154. This example is a diagram of the concatenated VPARS database structure that this tutorial follows.

In this database structure, records that should be used by all members of a department are loaded to a Department database. Records that should be used by all members of a group are loaded to a Group database. Records that are used by all members of a team are loaded to a Team database, and records used by only one user can be loaded to that user's read/write database. Each database level (Team, Group, and Department) can contain up to 128 minidisks.

As mentioned above, if you define an alternate set of disks for each Group database (for example), then a maintenance programmer can link read/write to a Group database and load a new set of records to it, while the regular TPF programmers are using the other set of Group disks for their testing. You would need an alternate set of disks for each Group.

Defining the Structure

Our first task is to define the minidisks for the VPARS databases in the VM CP directory. The User's read/write database will be defined in the User directory entries. The concatenated databases will be defined in a separate directory entry and linked read/only by the users. The directory entries for this example are defined in the VPARS sample directory entries that are listed later.

Next we need to create the configuration files for each level of VPARS database, the concatenation files that will be used for testing, and the maintenance of the different concatenation levels.

We will create four configuration files, one for each level of VPARS concatenation. All the configuration files will use the same TPF device and format tables.

1. User level configuration (USERCFG).

```
* Define a VPCONFIG file
CONFIG_TYPE *CFG
* Base VPARS minidisk address
BASE 1000
* Maximum VPARS minidisks
VPMD_max 128
* Default TPF device table name
TPFDEV TPFDEV1
```

2. Team level configuration (TEAMCFG).

```
* Define a VPCONFIG file
CONFIG_TYPE *CFG
* Base VPARS minidisk address
BASE 1080
* Maximum VPARS minidisks
VPMD_max 128
* Default TPF device table name
TPFDEV TPFDEV1
```

3. Group level configuration (GROUPECFG).

```
* Define a VPCONFIG file
CONFIG_TYPE *CFG
* Base VPARS minidisk address
BASE 1100
* Maximum VPARS minidisks
VPMD_max 128
* Default TPF device table name
TPFDEV TPFDEV1
```

4. Department level configuration (DEPTCFG).

```
* Define a VPCONFIG file
CONFIG_TYPE *CFG
* Base VPARS minidisk address
BASE 1180
* Maximum VPARS minidisks
VPMD_max 128
* Default TPF device table name
TPFDEV TPFDEV1
```

We will create three concatenation files to allow testing at any level of the concatenated VPARS database.

1. Concatenation at Team level. (TEAMCCT)

```

* Define a CONCAT file
CONFIG_TYPE *CCT
* Three concatenated databases
CONCAT TEAMCFG ,
      GROUPCFG ,
      DEPTCFG

```

2. Concatenation at Group level. (GROUPCCT)

```

* Define a CONCAT file
CONFIG_TYPE *CCT
* Two concatenated databases
Concat GROUPCFG DEPTCFG

```

3. Concatenation at Department level. (DEPTCCT)

```

* Define a CONCAT file
CONFIG_TYPE *CCT
CONCAT DEPTCFG

```

You could also create a concatenation of just the Team and Department levels or even reverse the order of concatenation. However, both of these would allow a developer to test against the wrong level of program and data records.

Testing with the Concatenation

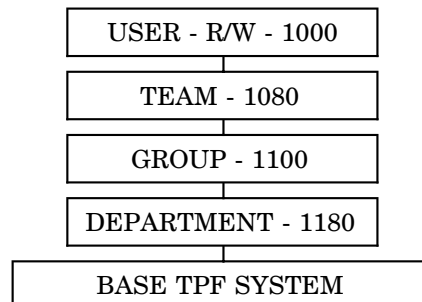
To prepare for a TPF test, a user would logon to VM and run an exec that would link to the current set minidisks for each level (Team, Group, and Department). “Concatenation Example 2” on page 155 is a sample exec that could be used to link to the VPARS database minidisks for users TEAMA1A and TEAMA1B.

The user could then test against any level in the concatenation. Most testing would probably be performed with the full concatenation.

To test against the full concatenation the user would issue:

```
VPOPEN CONCAT TEAMCCT
```

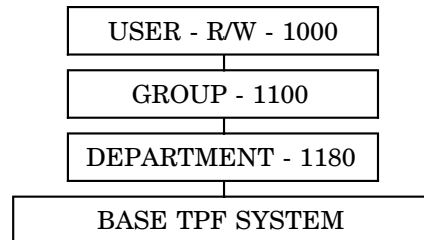
The search order for requested TPF records would be:



To test against only the Group and Department levels the user would issue:

```
VPOPEN CONCAT GROUPCCT
```

Records loaded to the TEAMCFG database would not be included in this structure. The disks for the TEAMCFG database may or may not be linked, but if they are not referenced in the concatenation used by VPOPEN, they will not be searched by VPARS. The order of search for requested TPF records would be:



Updating the Databases

A database maintenance group would be responsible for updating the VPARS databases and the execs used to link the database minidisks. A different exec is required for each team, or the exec could determine which disks to link based on the userid. This exec would link to the correct team and group disks for the user, and to the department disks (which are common for all users in the department).

To build a new database at the Team level, a maintenance user would link read/write to the alternate database for that team and read/only to the current Group and Department levels. (If a VPARS database is being used in a concatenation by other users, it can not be linked read/write and updated.) The programmer would then issue:

```
VPOPEN CONFIG TEAMCFG CONCAT GROUPCCT
```

This maintenance user could then clear this database and load new records to it for later use by all programmers. Note that the current Group and Department levels should be in the concatenation so that all current records are picked up by the TPF system that's reloading the Team records.

To build a new database at the Department level, a maintenance user would link read/write to the alternate Department database and issue:

```
VPOPEN CONFIG DEPTCFG
```

After a database level has been updated and tested, the prime and alternate minidisks for that database can be changed in the link exec used by the users. As new users logon and issue the link exec, they will pick up the new database disks for the updated level. Note that the target disk addresses used in the link are different, depending on whether the original or alternate disks are linked, but the "link as" addresses are always the same -- for example, the disks for the Group database are always linked in the 1100 range. Current users would have to close their VPARS database, detach the disk for the old level, and reissue the link exec to get the updated database.

You should consider that if you build a new database for a concatenation level, you may have to clear or rebuild the databases above that level. Records on the

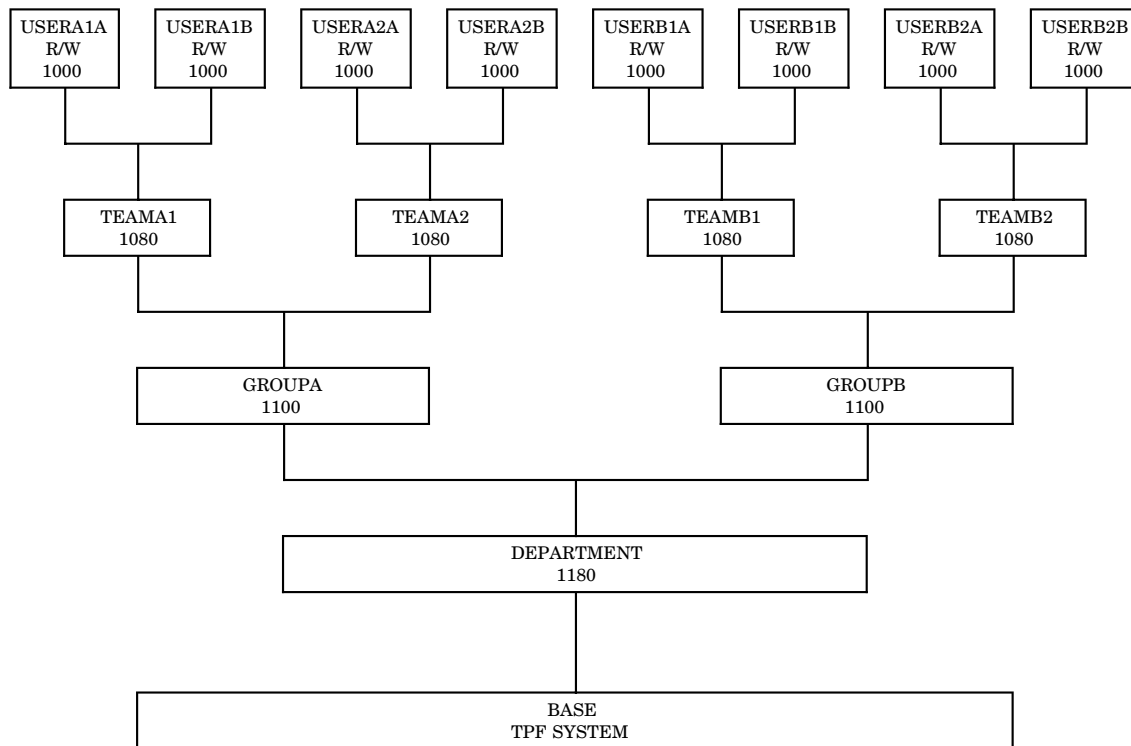
database you built may also be on a higher level database. VPARS will find the first copy of a record in the highest concatenation level, so if the higher-level databases are not cleared or rebuilt, VPARS may not find the records that you just loaded.

Minidisks for a level must always be linked with the same virtual device number by all users. If the disks are linked with a different virtual device number the VPARS database will not open.

Concatenation Example 1

This example shows the relationship of the different levels of VPARS database concatenation to the test users, and illustrates the search order for TPF records for each user. (Each box represents a different set of VPARS database disks. The starting disk address is shown. Each database could contain up to 256 disks, if all addresses ended with 00. This example assumes 128 disks will be enough, so addresses that end with 80 are used.)

The directory entries that allocate the minidisks for this configuration are in the VPARS sample directories. All of the minidisks below the user level are defined in the directory entry for user DEPARTMT.



Concatenation Example 2

This is a sample exec that could be used to link to the VPARS minidisks for users TEAMA1A and TEAMA2A.

```
/* Link to concatenated VPARS databases */
address command

/* The following conventions are used for concatenated database */
/* minidisk links. */
/* */
/* 1080-10FF - Team level database. */
/* 1100-117F - Group level database. */
/* 1180-11FF - Department level database. */

ecp = 'EXECIO * CP (STEM EMSG. STRING LINK'
final_rc = 0

call links 'DEPARTMT' 2080 1080 2 /* Primary TEAMA1 disks. */
call links 'DEPARTMT' 3100 1100 2 /* Alternate GROUPE disks. */
call links 'DEPARTMT' 2180 1180 4 /* Primary department disks. */
exit final_rc

links:
arg linkuser linkto linkas linkcnt .

linkto = x2d(linkto) /* Convert link addresses to */
linkas = x2d(linkas) /* decimal. */

do i = 0 to linkcnt - 1 /* Link VPARS database set. */
  ecp linkuser d2x(linkto+i) d2x(linkas+i) 'RR READ'
  if rc ^= 0 then do /* If link failed, */
    final_rc = rc /* set final return code and */
    say emsg.1 /* display error message. */
  end
end
return /* Return to main routine. */
```


Chapter 16. Directory Requirements

There are two sections to the VPARS directory sample.

The first section, starting with user TPFMAINT through user VPMDPOOL, defines userids and minidisks to support a normal VPARS TPF testing environment. In this section VPARS minidisks are defined on real disks VPARS1 and VPARS2 in userids TPFMAINT and VPMDPOOL. These minidisks are linked by test system users for testing.

The second section, starting with user DEPARTMT through user USERB2B, defines userids and minidisks to support a multilevel concatenated VPARS database environment. In this section VPARS minidisks are defined on real disks DEP001-DEP003 and DEP128-DEP131. The minidisks for concatenated VPARS data bases are defined in user DEPARTMT. The minidisks for each test user's read/write VPARS database are defined in that user's directory entry.

The standard CMS environment and the links to the TPF base system minidisks are set up in a profile named TPFTEST.

The VPARS minidisks must all be formatted with VPFMT before they can be used for testing.

Sample Directory Entries

* Sample directory entries for VPARS
* This profile provides all common resources that are
* required for a TPF test system user.

PROFILE TPFTEST

MACHINE XA

CONSOLE 01F 1052

SPOOL 00C 2540 READER *

SPOOL 00D 2540 PUNCH B

SPOOL 00E 1403 A

* Normal CMS minidisks

LINK MAINT 190 190 RR

LINK MAINT 19D 19D RR

LINK MAINT 19E 19E RR

* TPF base system disks

LINK TPFMAINT 260 260 RR

LINK TPFMAINT 261 261 RR

LINK TPFMAINT 262 262 RR

LINK TPFMAINT 268 268 RR

LINK TPFMAINT 269 269 RR

LINK TPFMAINT 2D6 2D6 RR

* TPF maintenance virtual machine

* This userid owns the VPARS database disks and

* is used to apply updates to the base TPF system

USER TPFMAINT TPF 2M 4M BG

MACHINE XA

ACCOUNT TPF SYSTEMS

CONSOLE 01F 1052

SPOOL 00C 2540 READER *

SPOOL 00D 2540 PUNCH B

SPOOL 00E 1403 A

LINK MAINT 190 190 RR

LINK MAINT 19D 19D RR

LINK MAINT 19E 19E RR

MDISK 191 3380 550 001 VMPK01 WR READ WRITE

* TPF system disks

MDISK 260 3380 000 885 TPF001 MR READ

MDISK 268 3380 000 885 TPF002 MR READ

MDISK 262 3380 000 885 TPF003 MR READ

MDISK 269 3380 000 885 TPF004 MR READ

MDISK 263 3380 000 885 TPF005 MR READ

MDISK 2D6 3380 000 885 GENF01 MR READ

* VPARS minidisks assigned to TPF test user VPTPF1

MDISK 1F00 3380 001 100 VPARS1 MW READ WRITE

MDISK 1F01 3380 001 100 VPARS2 MW READ WRITE

* VPARS minidisks assigned to TPF test user VPTPF2

MDISK 1F02 3380 101 100 VPARS1 MW READ WRITE

* Available formatted VPARS minidisks

MDISK 1F03 3380 101 100 VPARS1 MW READ WRITE

```

MDISK 1F04 3380 201 100 VPARS1 MW READ WRITE
MDISK 1F05 3380 201 100 VPARS2 MW READ WRITE
*   These link entries provide the ability to format
*   the VPARS disks that are defined in VPMDPOOL
LINK VPMDPOOL 1FA0 1FA0 MW
LINK VPMDPOOL 1FB0 1FB0 MW
LINK VPMDPOOL 1FA1 1FA1 MW
LINK VPMDPOOL 1FB1 1FB1 MW
* VPARS TPF test user 1
USER VPTPF1 VPARS 2M 4M G
  INCLUDE TPFTEST
  ACCOUNT TPF USER1
  MDISK 191 3380 551 001 VMPK01 WR READ WRITE
* VPARS database minidisks
LINK TPFMAINT 1F00 1F00 MW
LINK TPFMAINT 1F01 1F01 MW
*   VPARS TPF test user 2
USER VPTPF2 VPARS 2M 4M G
  INCLUDE TPFTEST
  ACCOUNT TPF USER2
  MDISK 191 3380 552 001 VMPK01 WR READ WRITE
* VPARS database minidisk
LINK TPFMAINT 1F02 1F00 MW
*   VPARS minidisk pool virtual machine
USER VPMDPOOL NOLOG 1M 1M G
  MDISK 1FA0 3380 451 050 VPARS1 MW
  MDISK 1FB0 3380 451 050 VPARS2 MW
  MDISK 1FA1 3380 501 050 VPARS1 MW
  MDISK 1FB1 3380 501 050 VPARS2 MW
*   Directory entries for the concatenation tutorial
*   User DEPARTMT contains all minidisk statements for
*   concatenated VPARS databases.
USER DEPARTMT
* (Department)
  MACHINE XA
  ACCOUNT TPF SYSTEMS
  CONSOLE 01F 1052
  SPOOL 00C 2540 READER *
  SPOOL 00D 2540 PUNCH B
  SPOOL 00E 1403 A
  LINK MAINT 190 190 RR
  LINK MAINT 19D 19D RR
  LINK MAINT 19E 19E RR
  MDISK 191 3380 553 001 VMPK01 WR READ WRITE
*   Primary VPARS minidisks for Department level   DEPARTMENT
  MDISK 2180 3380 0000 0485 DEP000 MR READ WRITE
  MDISK 2181 3380 0000 0485 DEP001 MR READ WRITE
  MDISK 2182 3380 0000 0485 DEP002 MR READ WRITE
  MDISK 2183 3380 0000 0485 DEP003 MR READ WRITE
*   Alternate VPARS minidisks for Department level
  MDISK 3180 3380 0000 0485 DEP128 MR READ WRITE

```

```

MDISK 3181 3380 0000 0485 DEP129 MR READ WRITE
MDISK 3182 3380 0000 0485 DEP130 MR READ WRITE
MDISK 3183 3380 0000 0485 DEP131 MR READ WRITE
*   Primary VPARS minidisks for group level          GROUPA
MDISK 2100 3380 0485 0100 DEP000 MR READ WRITE
MDISK 2101 3380 0485 0100 DEP128 MR READ WRITE
*   Alternate VPARS minidisks for Group level
MDISK 3100 3380 0485 0100 DEP001 MR READ WRITE
MDISK 3101 3380 0485 0100 DEP129 MR READ WRITE
*   Primary VPARS minidisks for Group level          GROUPB
MDISK 4100 3380 0485 0100 DEP002 MR READ WRITE
MDISK 4101 3380 0485 0100 DEP130 MR READ WRITE
*   Alternate VPARS minidisks for Group level
MDISK 5180 3380 0485 0100 DEP003 MR READ WRITE
MDISK 5181 3380 0485 0100 DEP131 MR READ WRITE
*   Primary VPARS minidisks for Team level          TEAMA1
MDISK 2080 3380 0585 0050 DEP000 MR READ WRITE
MDISK 2081 3380 0585 0050 DEP128 MR READ WRITE
*   Alternate VPARS minidisks for Team level
MDISK 3080 3380 0585 0050 DEP001 MR READ WRITE
MDISK 3081 3380 0585 0050 DEP129 MR READ WRITE
*   Primary VPARS minidisks for Team level          TEAMA2
MDISK 4080 3380 0635 0050 DEP000 MR READ WRITE
MDISK 4081 3380 0635 0050 DEP128 MR READ WRITE
*   Alternate VPARS minidisks for Team level
MDISK 5080 3380 0635 0050 DEP001 MR READ WRITE
MDISK 5081 3380 0635 0050 DEP129 MR READ WRITE
*   Primary VPARS minidisks for Team level          TEAMB1
MDISK 6080 3380 0585 0050 DEP003 MR READ WRITE
MDISK 6081 3380 0585 0050 DEP130 MR READ WRITE
*   Alternate VPARS minidisks for Team level
MDISK 7080 3380 0585 0050 DEP004 MR READ WRITE
MDISK 7081 3380 0585 0050 DEP131 MR READ WRITE
*   Primary VPARS minidisks for Team level          TEAMB2
MDISK 8080 3380 0635 0050 DEP003 MR READ WRITE
MDISK 8081 3380 0635 0050 DEP130 MR READ WRITE
*   Alternate VPARS minidisks for Team level
MDISK 9080 3380 0635 0050 DEP004 MR READ WRITE
MDISK 9081 3380 0635 0050 DEP131 MR READ WRITE
USER USERA1A
* (Group A Team 1 User A)
INCLUDE TPFTEST
ACCOUNT CONCAT USERA1A
MDISK 191 3380 554 001 VMPK01 WR READ WRITE
*   VPARS database minidisks
MDISK 1000 3380 0685 0050 DEP000 MR READ WRITE
MDISK 1001 3380 0685 0050 DEP128 MR READ WRITE
USER USERA1B
* (Group A Team 1 User B)
INCLUDE TPFTEST
ACCOUNT CONCAT USERA1B

```

```

MDISK 191 3380 552 001 VMPK01 WR READ WRITE
*   VPARS database minidisks
MDISK 1000 3380 0685 0050 DEP001 MR READ WRITE
MDISK 1001 3380 0685 0050 DEP129 MR READ WRITE
USER USERA2A
* (Group A Team 2 User A)
  INCLUDE TPFTEST
  ACCOUNT CONCAT USERA2A
  MDISK 191 3380 552 001 VMPK01 WR READ WRITE
*   VPARS database minidisks
MDISK 1000 3380 0735 0050 DEP000 MR READ WRITE
MDISK 1001 3380 0735 0050 DEP128 MR READ WRITE
USER USERA2B
* (Group A Team 2 User B)
  INCLUDE TPFTEST
  ACCOUNT CONCAT USERA2B
  MDISK 191 3380 557 001 VMPK01 WR READ WRITE
*   VPARS database minidisks
MDISK 1000 3380 0735 0050 DEP001 MR READ WRITE
MDISK 1001 3380 0735 0050 DEP129 MR READ WRITE
USER USERB1A
* (Group B Team 1 User A)
  INCLUDE TPFTEST
  ACCOUNT CONCAT USERB1A
  MDISK 191 3380 558 001 VMPK01 WR READ WRITE
*   VPARS database minidisks
MDISK 1000 3380 0685 0050 DEP003 MR READ WRITE
MDISK 1001 3380 0685 0050 DEP130 MR READ WRITE
USER USERB1B
* (Group B Team 1 User B)
  INCLUDE TPFTEST
  ACCOUNT CONCAT USERB1B
  MDISK 191 3380 559 001 VMPK01 WR READ WRITE
*   VPARS database minidisks
MDISK 1000 3380 0685 0050 DEP004 MR READ WRITE
MDISK 1001 3380 0685 0050 DEP131 MR READ WRITE
USER USERB2A
* (Group B Team 2 User A)
  INCLUDE TPFTEST
  ACCOUNT CONCAT USERB2A
  MDISK 191 3380 560 001 VMPK01 WR READ WRITE
*   VPARS database minidisks
MDISK 1000 3380 0735 0050 DEP003 MR READ WRITE
MDISK 1001 3380 0735 0050 DEP130 MR READ WRITE
USER USERB2B
* (Group B Team 2 User B)
  INCLUDE TPFTEST
  ACCOUNT CONCAT USERB2B
  MDISK 191 3380 561 001 VMPK01 WR READ WRITE
*   VPARS database minidisks
MDISK 1000 3380 0735 0050 DEP004 MR READ WRITE

```

MDISK 1001 3380 0735 0050 DEP131 MR READ WRITE

Chapter 17. VPARS CP Messages

This section lists the VPARS CP messages with explanations, and user actions. Informational messages are usually issued without the message number. The full message number is RVPxxxnnnt, where:

xxx identifies the module that issued the message, such as PRC for module RVPPRC.

nnn is the message "number". (VPARS also issues some IBM error messages. These messages will have a 4-digit numeric message number *nnnn*. Explanations of these messages can be found in the IBM System Messages and Codes Reference.) Note, however, that some of VPARS CP module names end in a digit, and this digit is not part of the message number. Only the three characters of the message number (before the I, E, or W message type) should be used to look up an error message.

t is the message type:

- I for informational messages
- W for warnings
- E for errors
- S for severe errors
- R for messages that require a response (such as YES or NO)

A typical message number is RVPDBM000S. This message was issued by VPARS module RVPDBM, and is message number 000S (the first message described below).

000S VPARS record directory error, code=*n*

Explanation: A logical error was found in the VPARS directory index. This error may occur for the following reasons:

- The VPARS control record was overlaid in free storage.
- A record directory index IOBLOK or virtual buffer was overlaid.
- A database disk was overwritten by something other than VPARS (such as DDR).
- A hardware I/O error occurred on a database disk.

n is an internal code that identifies the type of error in the record directory.

System Action: The I/O request is rejected with a Command Reject.

User Action: Notify your system programming group. If the error persists, the VPARS database may have been damaged, and you may have to clear the database.

System Programmer Action: Note the complete error message number and text of the message and contact Virtual Software Systems.

003E VPARS open failed

Explanation: VPARS was unable to open the VPARS database. This message was preceded by a message describing the reason the database could not be opened.

User Action: Correct the error and reissue the VPOPEN command to open the database.

004E VPARS database minidisk *vdev* is not linked.

Explanation: There is not a minidisk at virtual device address *vdev*.

User Action: VPARS may be trying to open your database with a different configuration file than the one you expect. VPQUERY CONFIG will show the name and definition of your default configuration file. If the configuration name is correct, you should link a VPARS database disk or system pool disk, or define and format a system T-disk at the correct address and reissue the VPOPEN command.

005E Device at address *vdev* is not a disk

Explanation: A device in the VPARS database address range is not a disk.

vdev is the virtual device number in error.

User Action: Make sure that VPARS is using the correct configuration file. Detach the device identified by *vdev*, ensure that the devices in the database address range are correct, and reissue the VPOPEN command to open the database.

006I VPARS database is open

Explanation: The VPARS database has been successfully opened.

007E VPARS disks are out of sequence. *vdev* is *nn* but should be *mm*

Explanation: An error has been detected in the minidisk sequence of a VPARS database. *vdev* is the virtual address of the disk in error. *nn* is the sequence number that was written to that minidisk when it was last part of a VPARS database. *mm* is the sequence number that should be found on the disk at that address.

User Action: Correct the error in the sequence of the disks, and reissue the VPOPEN command to open the database, or use the VPOPEN CLEAR command to open and clear the database.

008E Only *nn* VPARS disks are linked, but *mm* disks are active in the database

Explanation: One or more minidisks that were part of a database are not linked. *nn* is the number of minidisks linked. *mm* is the number of minidisks that contain active TPF data records.

User Action: Make the missing minidisk(s) available to the virtual machine and reissue the VPOPEN command to open the database, or use VPOPEN CLEAR to open and clear the database. If you are not clearing the database, you cannot open it unless you have links to all of the minidisks that contain data.

009E VPARS database is not open

Explanation: A VPARS function has been requested that requires the VPARS database to be open, but the database is closed.

User Action: Open the VPARS database and reenter the request.

011W VPARS database is open

Explanation: VPOPEN has been requested for a database that is already open.

User Action: A database is already open. If you are trying to open a different VPARS database, you must first close your current database.

012I VPARS database is closed

Explanation: The VPARS database has been closed either in response to a VPCLOSE request, or a LOGOFF command.

013E *option* is an invalid option

Explanation: A VPARS command has been entered with an option that is not valid. *option* is the invalid option.

User Action: Check the allowable options for the command you are issuing. Reenter the VPARS command with a valid option.

014I VPARS database=*vdev-vdev*, Active=*nn*, %Full=*nn*, Sync=*nn*, Sync I/O=*nn*

Explanation: This message is part of the response to VPQUERY STATUS.

VPARS database=*vdev-vdev*

The virtual device number range (address range) of the linked database disks.

Active=*nn*

The number of database minidisks that contain active TPF data records.

Full=*nn*%

The percent full of your VPARS database.

Sync=nn

The number of internal calls to the database synchronization routine. Normally, this value is not meaningful to the general VPARS user.

Sync I/O=nn

The number of I/O requests performed by the database synchronization routine. Normally, this value is not meaningful to the general VPARS user.

015I **** Database key *dbkey*, User *userid*, Date *date* at time *version* ****

015I **** Database key *dbkey*, with *n* users, Date *date* at time *version* ****

Explanation: This message, part of the VPQUERY STATUS response, shows the database key, userid, date, time, and the database version. If more than one user has the database open, the number of users is shown instead of the userid. VPARS version 3.2 supports databases from VPARS version 2.1, along with database versions 3.2 and 3.3 from VPARS version 3.2. The key is an eight-character hexadecimal number (based on the real location of the first minidisk in the database) used to uniquely identify each open database in the system.

016I Blks=*nn*, Actv=*nn*, II=*nn*, DI=*nn*, RD=*nn*, DB=*nn*, Locked=*nn*

Explanation: This message, part of the response to VPQUERY STATUS, shows the numbers of various kinds of blocks in the VPARS database. All figures, except for the Locked value, represent 4096-byte data blocks on the database.

Blks=nn

The capacity of the database, in 4096-byte blocks.

Actv=nn

The number of 4096-byte blocks in use on the database. This number includes the following four values, plus a 100-block buffer that is used to write currently active blocks when the database becomes 100 percent full, plus a few other blocks that are not explicitly listed here.

II=nn

The number of blocks in use on the database for index index records.

DI=nn

The number of blocks in use on the database for directory index records.

RD=nn

The number of blocks in use on the database for record directories.

DB=nn

The number of blocks in use on the database for TPF data records.

Locked=nn

The number of 4096-byte buffers locked in storage for active I/O to the VPARS database.

017I TPF records: Total=*nn*, Added=*nn*, Found=*nn*, Updated=*nn*

Explanation: This message, part of the response to VPQUERY STATUS, shows TPF record information from the VPARS database.

Total=*nn*

The number of TPF data records (of all sizes) on the VPARS database.

Added=*nn*

The number of TPF data records that were written to the VPARS database for the first time.

Found=*nn*

The number of TPF data records that were read from the VPARS database.

Updated=*nn*

The number of TPF data records that were found on the VPARS database for a write request, and were updated.

018I There is no current activity on any selected database

Explanation: This message is issued in response to a VPQUERY ACTIVITY command if no activity was found on any requested database.

019I TPF requests=*nn*, Active=*nn*, /sec=*nn*

Explanation: This message, part of the response to VPQUERY STATUS, shows TPF I/O requests and activity.

TPF requests=*nn*

The number of I/O requests that have been issued by the TPF guest.

Active=*nn*

The number of guest I/O requests currently being processed.

/sec=*nn*

The number of guest I/O requests that were issued in the past second.

020I Status interval=*nn*, Buffers: Minimum=*nn*, In use=*nn*, Maximum=*nn*

Explanation: This message, part of the response to VPQUERY STATUS, shows the status interval and buffer counts. Buffers are 4096-byte blocks of storage used to read and write data and directory blocks from and to the database.

Status interval=*nn*

The number of minutes between automatic displays of VPQUERY STATUS.

Buffers: Minimum=*nn*

The minimum number of buffers that VPARS will use for database I/O.

In use=*nn*

The number of buffers VPARS is currently using.

Maximum=*nn*

The maximum number of buffers that VPARS will use.

021E Option or parameter missing for VPARS command

Explanation: A required parameter was not specified when the VPARS command was entered.

User Action: Reenter the VPARS command specifying the required parameter.

022E *option* option conflicts with a previously specified option

Explanation: An option was entered that conflicts with an option that was already specified.

User Action: Check the syntax of the command and any mutually exclusive options.

023I Status interval set to *nn* minutes

Explanation: This message is issued when you change the interval between automatic displays of VPQUERY STATUS.

025E Loosely coupled database is already open, checkpoint option was ignored

Explanation: This message is issued when you open a database with the COUPLED and CHECKPOINT options, and one or more users already have the same database open. You are allowed to join the already-open database as an additional user, but it is not opened at the checkpoint you specified.

User Action: If you need to open (or clear) the database to a checkpoint, there must be only one read/write user. All other users must close the database first.

026I VPARS [database | checkpoint] clear complete

Explanation: The VPARS database clear or checkpoint clear has been done as requested.

027S Invalid VPARS relative block=*nnnn*, IOT=*address*, Caller=*module*, Offset=*nnn*

Explanation: A relative block address outside of the VPARS database extents has been passed to the block convert routine. This is either an internal error, or the data on the VPARS database disks has been corrupted.

nnnn

The relative block number passed.

IOT=*address*

The real storage address of the VPARS I/O task block.

Caller=*module*

The VPARS CP module that passed the block number.

Offset=*nnn*

The offset of the call to the block convert program within the above VPARS module.

System Action: The virtual machine is reset and placed in a disabled wait state.

User Action: Notify your system programming group.

System Programmer Action: Note the complete error message number and text of the message and contact Virtual Software Systems.

028E I/O error, VADDR=*vdev*, RADDR=*address*, IORCASC=*n*, RELR=*re*, CCHHR=*cchhr*, RCYL=*nn*, User=*userid*

Explanation: An I/O error has occurred on a VPARS database disk. This can be caused by an improperly formatted disk, a defective disk, or a hardware I/O error. This message will be issued to the user and to the VM system operator.

VADDR=*vdev*

The virtual device number (address) of the disk that had the error.

RADDR=*rdev*

The real device number of the disk that had the error.

IORCASC=*n*

The IORCASC returned from the I/O request.

RELR=*re*

The relative record number of the record on a VPARS minidisk.

CCHHR=*cchhr*

The cylinder, head, and record for the error.

RCYL=*nn*

The real cylinder number on the disk.

User=*nn*

The VPARS user who had the error.

User Action: If the error persists, notify your system programming group.

System Programmer Action: Determine whether there have been any real I/O errors on the disks involved. If not, note the complete error message number and text of the message and contact Virtual Software Systems.

029W VPARS database is *nn* percent full, user=*userid*

Explanation: This message is issued for every percentage point the VPARS database fills from 90 to 100 percent. The message is sent to the user and to the VM system operator. If the users performing TPF testing are dialed in to the test system, or not at the prime CRAS, they will not see this message. The VM system operator should notify the system programming group of any occurrences of this message.

System Action: At 100 percent full, the virtual machine is placed in a disable wait state, with a PSW instruction counter of "00000DBF"

User Action: Add a VPARS formatted minidisk to the database to provide additional capacity. See the VPLINK and VPADD commands for more information.

030I *vdev* was added to your VPARS database

Explanation: An additional disk has been added to the VPARS database with VPADD. *vdev* is the virtual device number of the added disk.

031E *vdev* is not available

Explanation: VPADD was entered, but a minidisk does not exist at the next available device number *vdev* in the database address range.

User Action: Link a VPARS formatted disk at the required virtual device number, and reenter the request. If your installation has defined VPARS pool minidisks, you may be able to use the VPLINK command to obtain an additional minidisk. See the VPLINK command for more information, and backup considerations when using pool minidisks.

033I VPARS checkpoint *nn* set

Explanation: A checkpoint has been set as requested. *nn* is the new checkpoint number. You can clear the database to any checkpoint when desired.

035E Control key on VPARS disk *vdev* does not match the base disk

Explanation: During the open validation of a multiple minidisk VPARS database, a minidisk was found to have a control key that does not match the key on the first minidisk in the database. *vdev* is the virtual device number of the minidisk that contains the wrong key.

Probable causes are:

- The minidisk in error has been formatted with the 'VPFMT' command since it was last part of the database.
- Another VPARS user may have linked to the minidisk and used it in a database.

User Action: If the data on the minidisk has been overlaid, the VPARS database is no longer usable. It can be opened with the CLEAR option, or all disks in the database can be formatted.

036E PTV keypoint error

Explanation: During PTV mode processing, a record was read from the VPARS database that should have contained PTV or keypoint records, but the record header was not correct.

System Programmer Action: Note the complete error message number and text of the message and contact Virtual Software Systems.

037I PTV mode entered

Explanation: This message is issued when the VPSET PTV ON command is entered.

038E PTV mode is not active

Explanation: A VPSET PTV OFF command was entered when PTV mode was not active.

User Action: None.

039I VPARS flags: PTV=*onoff*, Checkpoint=*nn onoff*, Clear=*mode*

Explanation: This message, part of the response to VPQUERY STATUS, shows any special options in effect.

PTV=*onoff*

Whether PTV mode is on or off.

Checkpoint=*nn onoff*

The current checkpoint number, followed by ON if any checkpoints are set, and OFF if none are set (or if checkpointing was turned off after checkpoints were set)

Clear=*mode*

Normal or Restricted mode for VPCLEAR.

040E VPARS disk *vdev* was not formatted with the current VPFMT command

Explanation: A control record header in a VPARS minidisk does not match the version of VPARS installed.

User Action: Format the base VPARS minidisk with the current level of the VPFMT command.

System Programmer Action: Ensure that the VPFMT module that was built during VPARS installation has been copied to a minidisk accessible to the VPARS users.

044I TPF records: PTV=*nn*, Checkpoint=*nn*

Explanation: This message, part of the response to VPQUERY STATUS, shows if any PTV or checkpoint records are on the database. If not, this message is not issued.

PTV=*nn*

The number of keypoint and PTV records on the database.

Checkpoint=*nn*

The number of TPF records that were on the database when the last checkpoint was set, that have been updated.

045I VPARS I/O=*nn*, Reads=*nn*, Writes=*nn*, Active=*nn*, /sec=*nn*

Explanation: This message, part of the response to VPQUERY STATUS, shows VPARS database I/O activity.

I/O=*nn*

The total number of I/O operations done by VPARS to satisfy requests for TPF records and synchronize the record directory since the last open or clear for the database.

Reads=*nn*

The number of I/O operations that were reads.

Writes=*nn*

The number of I/O operations that were writes.

Active=*nn*

The number of I/O operations currently active. Active refers to I/O requests that are currently being processed by VPARS for this database.

/sec=*n*

The number of I/O operations performed in the last second.

046W Active VPARS disk *vdev* is being detached.

Explanation: The minidisk at address *vdev* is being removed from an open database. See message number B0047I.

047I VPARS disk *vdev* has been removed from the database

Explanation: A CP detach or VPREMOVE command was entered for the minidisk at address *vdev*. The minidisk was part of an open database.

System Action: All minidisks in the database with an address equal to or higher than the detached minidisk are removed from the database.

User Action: None.

048E Error return code from CP LINK

Explanation: An internal CP link command was done as part of VPLINK, and received a nonzero return code.

User Action: Notify your system programming group.

System Programmer Action: Note the complete error message number and text of the message and contact Virtual Software Systems.

049E Error reading CP directory

Explanation: A nonzero return code was returned to the VPLINK command from HCPUDR.

User Action: Notify your system programming group.

System Programmer Action: Note the complete error message number and text of the message and contact Virtual Software Systems.

050E No pool disks of the requested type and size are available

Explanation: All minidisks of the type requested from the pool are in use, or no minidisks of the type requested are defined in the CP directory entry of the pool minidisk owner userid. Type refers to the disk type (3380, 3390, etc.).

User Action: Reenter the VPLINK request with a device type supported by your system configuration, or specify the SIZE 0 parameter to link a disk of any available size.

051E No addresses are available in the disk range *vdev-vdev*

Explanation: This message is issued by VPLINK. Your VPARS database address range is full; you have the maximum number of VPARS minidisks defined. *vdev-vdev* shows the database range defined for your configuration.

User Action: If you need to use a larger database, your system programming group can define a configuration with more minidisks in the database range.

054E Invalid number *xx* requested

Explanation: A VPLINK command was made for an invalid number of minidisks, or *xx* is not a number.

User Action: Enter the request with a valid number.

055E Unsupported TPF [read | write] CCW chain, VDEVNO = *vdev*

Explanation: A Start IO or Start Subchannel instruction was issued by TPF for a CCW chain that VPARS could not decode. *vdev* is the virtual device address in the I/O request.

System Action: If there are no write CCWS in the chain, it is processed against the real TPF disks. If there are write CCWS in the chain, channel end, device end, unit check, and channel program check are reflected to your virtual machine.

User Action: Notify your system programming group.

System Programmer Action: If the chain is a valid TPF CCW chain for your installation, contact Virtual Software Systems and provide a copy of the CCW chain.

056E **CCW=*ccw***

Explanation: This message is used to print the CCWs when an unsupported CCW chain is detected, in conjunction with the above message.

057E **Checkpoint control record ID error, record=*relr***

Explanation: The control record at relative record number *relr* for a checkpoint open or clear did not have the correct identifier. The record may have been overlaid, or the data on the minidisk may have been corrupted.

System Action: The database is not opened or cleared.

User Action: Notify your system programming group. If you can recreate the checkpointed data, clear the database, reload the data, set a new checkpoint and continue testing. You can continue testing with the VPARS database as it is, or you can clear the VPARS database and continue.

System Programmer Action: Note the complete error message number and text of the message and contact Virtual Software Systems.

058E **Add request exceeds number of disks defined in database range**

Explanation: A VPADD request was added with a number larger than the number of minidisks defined in the configuration for the open database.

User Action: Reissue the command with a valid number or 'ALL'.

059E ***vdev* has not been formatted with the VPFMT command**

Explanation: A disk in the VPARS address range is not a VPARS database minidisk.

User Action: Detach or redefine the device to another virtual address and reissue the request. If the disk should be a VPARS database minidisk, it has not been correctly formatted or has been changed.

060W **PTV mode is already active**

Explanation: A VPSET PTV ON command was entered while PTV mode is active.

User Action: None.

061E VPARS pool disk userid *userid* was not found in the CP directory

Explanation: The userid defined (in the VPARS system configuration file) to contain the minidisk definitions for VPARS pool disks is not in the CP directory.

User Action: Notify your system programming group.

063E No disks were removed, there are *nn* inactive disks

Explanation: A request was made to remove minidisks from the database. Either no minidisks were inactive, or the number specified to remove was greater than the number of inactive minidisks. Active minidisks will not be removed by the VPREMOVE command.

User Action: If you need to remove one or more minidisks, use the CP DETACH command.

064S Paging error, return code = *nn*, IOT = *address*, VMDBK = *address*, VPAGE = *nnnn*, return = *address*

Explanation: An error occurred during system paging.

User Action: Notify your system programming group.

System Programmer Action: Note the complete error message number and text of the message and contact Virtual Software Systems.

065W Database is already open, clear option was ignored

Explanation: This message is issued when you open a database with the COUPLED and CLEAR (or CLEAR CHECKPOINT) options, and one or more users already have the same database open. You are allowed to join the already-open database as an additional user, but it is not cleared.

User Action: If you need to clear the database, there must be only one read/write user. All other users must close the database first.

067E TPF device table is not available

Explanation: The configuration used during VPOPEN does not specify a device table.

System Action: VPARS does not open the database.

User Action: Check with your system programming group. You may want to use the DEVTBL option of VPOPEN to specify a device table.

068E Restricted function

Explanation: You have issued a command for which you are not authorized.

User Action: Notify your system programming group.

075E VPARS pool disk userid *userid* is in use

Explanation: The CP directory entry for the VPMDISKS userid is in use by the system. This directory entry is required in order to link a requested pool disk. VPARS will make ten retry attempts to access the directory entry before issuing this message.

User Action: Retry the command. If the error persists, notify your systems programming group.

System Programmer Action: Determine why VPARS received a nonzero return code from HCPUDROP, while trying to open the minidisk userid's directory entry.

076E Record ID check *xx*, record=*relr*

Explanation: A VPARS record read from the database at relative record *relr* did not have the expected record identifier *xx*. There may be an error in the database, or a database minidisk may have been corrupted.

User Action: Notify your system programming group. Clear and restore the database if possible.

System Programmer Action: Notify Virtual Software Systems of the error.

078E VPARS database is read only

Explanation: The VPARS database disks are linked read/only, or the database was opened read/only; VPARS cannot perform certain functions such as clearing the database. Also, TPF records will not be written to the database.

User Action: If you are trying to open a database, you must either have write links to the database disks, or you must open the database read/only. If your database is open read/only, you cannot clear it, set checkpoints, or write any records to it.

079I Database was last cleared on *date at time*

Explanation: This message, part of the response to VPQUERY STATUS, shows when the database was last cleared. This message is not issued if the database has never been cleared.

080I **Current checkpoint was taken on *date* at *time***

Explanation: This message, part of the response to VPQUERY STATUS, shows when the last checkpoint was set. This message is not issued if no checkpoints have been set.

081E **Cannot open to checkpoint *xx*, only *yy* checkpoints have been set**

Explanation: The checkpoint open cannot take place; the checkpoint you are requesting has not been set.

User Action: You can perform a normal database open, or open the database to a checkpoint that has been set.

082E **Checkpoint chain ends at checkpoint *nn***

Explanation: There is a database error in the checkpointed control records. The checkpoint open cannot take place to any number higher than the one indicated in the message.

System Action: The open does not complete.

User Action: Notify your system programming group. Retry the checkpoint open to a smaller checkpoint number, or try a normal open followed by a clear to a checkpoint.

System Programmer Action: Note the complete error message number and text of the message and contact Virtual Software Systems.

083E **Checkpoint *xx* requires *yy* disks. Only *zz* disks are available**

Explanation: There were *yy* active VPARS database minidisks when checkpoint *xx* was set. These same minidisks must be linked in order to open to checkpoint *xx*. However, there are only *zz* minidisks currently linked.

System Action: The open does not complete.

User Action: Obtain links to all the database disks that were present when the checkpoint was set, or retry the checkpoint open to a smaller checkpoint number.

084E **Cannot open to checkpoint *xx*, only 1 checkpoint has been set**

Explanation: The checkpoint open cannot take place; the checkpoint you are requesting has not been set.

System Action: The checkpoint open does not complete.

User Action: You can perform a normal database open, or open the database to checkpoint 1.

085E Checkpoints above *xx* require *yy* disks, only *zz* disks are available

Explanation: There were *yy* active database minidisks when checkpoint *xx* was set, but there are only *zz* database minidisks currently linked. The database cannot be opened to any checkpoint numbered *xx* or above until the missing minidisks are linked.

System Action: The checkpoint open does not complete.

User Action: Obtain links to all the database minidisks that were present when the requested checkpoint was set, or retry the checkpoint open with a smaller checkpoint number.

086E Checkpoint forward pointers not available until database has been cleared

Explanation: Forward pointers through the chain of checkpointed control records will not be set up until the database is cleared and new checkpoints are set. This message means that the software is at a later level than the VPARS database.

System Action: The checkpoint open does not complete.

User Action: You will not be able to perform a VPOPEN CHECKPOINT until the database is cleared and new checkpoints are set. You might be able to perform a normal database open followed by a clear to checkpoint.

087E Maximum number of VPARS users already active

Explanation: The number of users with open VPARS databases has reached the limit defined in the VPARS system configuration, or the limit set with the VPSET MAXUSERS command.

System Action: Your VPARS database is not opened.

User Action: Check with your system programming group. You will not be able to open a VPARS database until the limit is changed, or the number of VPARS users falls below the limit.

088E VPARS database at *vdev* is open for output by *nn* users

Explanation: The VPARS database whose base minidisk address is *vdev* is open for output by *nn* other users. You cannot open this database for input. You cannot open it for output unless you specify the COUPLED option for a loosely-coupled test. See the VPOPEN command for more information.

System Action: Your VPARS database is not opened.

User Action: If you are trying to open the correct database, determine why other users have the database open. If you are trying to open the database in loosely-coupled mode, all users must specify the COUPLED option during VPOPEN.

090I There are no active VPARS users

Explanation: This message may be issued in response to a VPQUERY command that lists all active VPARS users.

091E Cannot clear to checkpoint *nn*, only *mm* checkpoints have been set

Explanation: You are trying to clear to a checkpoint that has not been set.

User Action: Clear to a checkpoint that has been set.

092E Cannot clear to checkpoint *nn*, only 1 checkpoint has been set

Explanation: You are trying to clear to a checkpoint that has not been set.

User Action: Clear to a checkpoint that has been set.

094E VPARS database at *vdev* is open by *nn* other users

Explanation: The VPARS database whose base minidisk address is *vdev* is already open by other users. You cannot open this database for output.

System Action: Your VPARS database is not opened.

User Action: If you are trying to open the correct database, determine why other users have the database open.

095E VPARS database at *vdev* is open by user *userid*

Explanation: The VPARS database whose base minidisk address is *vdev* is already open by another user. You cannot open this database for output.

System Action: Your VPARS database is not opened.

User Action: If you are trying to open the correct database, determine why another user has the database open.

096E VPARS database at *vdev* is open for output by user *userid*

Explanation: The VPARS database whose base minidisk address is *vdev* is open for output by another user. You cannot open this database for input. You cannot open it for output unless you specify the COUPLED option for a loosely-coupled test. See the VPOPEN command for more information.

System Action: Your VPARS database is not opened.

User Action: If you are trying to open the correct database, determine why another user has the database open. If you are trying to open the database in loosely-coupled mode, all users must specify the COUPLED option during VPOPEN.

098E Invalid request, database has *nn* users

Explanation: Certain functions, such as Clear or Checkpoint clear, are not allowed during a loosely-coupled test. These functions would change the database that the other users are using, and it is not possible to coordinate the changes with the other users.

System Action: The function is not performed.

User Action: If you need to perform the function, you must have all other users close the database first.

099E VPARS database disk *vdev* is linked read/only

Explanation: You cannot open a database for output if any of the database disks are linked read/only.

System Action: The VPOPEN command is not performed.

User Action: Either use the RONLY option of VPOPEN to open the database read/only, or obtain write links to all the disks in the database.

100E CONFIG_TYpe statement missing.

Explanation: The first record of a VPARS file (VPSYSTEM, Configuration, Concatenation or Device table) being read is not the CONFIG_TYpe keyword.

User Action: Correct the file and re-issue the command or function.

System Action: The command or function is not executed.

101E Statement invalid for this configuration file.

Explanation: A keyword was encountered which is not part of the valid keywords for the type of configuration file being read, as defined by the CONFIG_TYpe keyword.

User Action: Correct the file and re-issue the command or function.

System Action: The command or function is not executed.

102E UWORD value *nnn* is invalid. Contact VSSI.

Explanation: This is an internal error.

System Action: The command or function is not executed.

System Programmer Action: Note the complete error message number and text of the message and contact Virtual Software Systems.

103E Duplicated CONFIG_TYpe statement.

Explanation: Only one CONFIG_TYpe statement is allowed per file.

User Action: Delete the duplicated statement.

System Action: The command or function is not performed.

104E Mixed statements in file.

Explanation: Some statements do not belong to the set of keywords supported by this file, as defined by the CONFIG_TYpe keyword.

System Action: The command or function is not performed.

User Action: Remove the unsupported statements and re-issue the command or function.

106E Maximum of 16 concatenation names exceeded.

Explanation: The maximum number of configurations which can be specified in one CONCAT file is 16. Whether they are defined with one or multiple CONCAT keywords, the total number is limited to 16.

User Action: Reduce the number of configuration names in the concatenation file.

System Action: The command or function is not executed.

107E Maximum of 253 device ranges exceeded.

Explanation: 253 is the maximum number of ranges which can be specified in a Device table.

User Action: Reduce the number of device ranges.

System Action: The command or function is not executed.

108E Device range overlap detected.

Explanation: In a device table, the starting address of a range must be greater than the ending address of the previous range, and the ending address must be lower than the starting address of the following range.

User Action: Correct the device address ranges.

System Action: The command or function is not executed.

109E Invalid data in configuration file *cfgname cfgtype*

Explanation: The data read from the configuration definition, concatenation definition, TPF device table definition, or TPF format table definition named *cfgname cfgtype* was invalid.

System Action: The command or function is not executed.

System Programmer Action: Note the complete error message number and text of the message and contact Virtual Software Systems.

110E VPARS Component ID block unavailable.

Explanation: This is, most likely, a VPARS internal error.

System Action: The command or function is not executed.

System Programmer Action: Note the complete error message number and text of the message and contact Virtual Software Systems.

111E *cfgname cfgtype* file was not found in system

Explanation: A VPARS configuration definition, concatenation definition, TPF device table definition, or TPF format table definition named *cfgname cfgtype* was not found in the system. This may be a name that was requested on a VPOPEN or VPQUERY command, a system or user default name.

User Action: If you are specifying a configuration, concatenation, device table, or format table name on the command, make sure you have the correct name.

System Programmer Action: If the a file with the requested name is present on the VSSI parm disk(s), note the complete error message number and text of the message and contact Virtual Software Systems. If the file is not present, you may need to create it.

112E No *cfgtype* file was found in system for userid *userid*

Explanation: A default configuration name or TPF device table name was not defined in the system for this userid, and none was specified on the command.

User Action: Check with your system programming group. You may be able to specify a configuration name or a TPF device table name on the command.

System Programmer Action: If default configuration or TPF device table names should be defined for the user, or system-wide, you will have to create the VPARS system defaults file.

113E Default concatenation file was not found

Explanation: You specified 'NOCONFIG' on VPOPEN, and a default concatenation file was not found in the system. Therefore, there is nothing to open.

User Action: Check with your system programming group.

System Programmer Action: Create a default concatenation, and update the system defaults file, if required.

114E VPARS system initialization failed

Explanation: VPARS system initialization did not complete. This message should have been preceded by another message identifying the reason for the failure.

User Action: Check with your system programming group.

System Programmer Action: If you cannot determine the cause of the problem, note the complete error message number and text of the message and contact Virtual Software Systems.

121E There is no database open at level *nn*

Explanation: The Level keyword was used in a VPQUERY command, but a database is not open at level *nn*. If you did not specify a userid, the VPQUERY is processed against your open databases.

User Action: Specify a level number that corresponds to an open database.

122E There is no database open with key *dbkey*

Explanation: The Key parameter was used in a VPQUERY command, but a database is not open with key *dbkey*. If you did not specify a userid or ALL, the VPQUERY is processed against your open databases.

User Action: Specify a key for a database that is open.

123E There is no database open with configuration name *cfgname*

Explanation: The Config (or Cfg) parameter was used in a VPQUERY command, but a database is not open with configuration name *cfgname*. If you did not specify a userid, the VPQUERY is processed against your open databases.

User Action: Specify a configuration name for a database that you or the requested user has open.

129E Requested user *userid* is not logged on

Explanation: The *Userid* parameter was used in a VPQUERY command, but the user is not logged on to the VM system.

User Action: Specify a *userid* for a user who is logged on.

143E *vdev* is not a disk

Explanation: A VPQUERY KEY command was issued specifying a virtual device number *vdev* that is not a disk. A database key is only meaningful for VPARS database minidisks.

User Action: Specify the correct virtual device number.

144E VPARS base disk *vdev* was not found

Explanation: A VPQUERY KEY command was issued specifying a virtual device number *vdev* and the disk was not linked. A database key is only meaningful for VPARS database minidisks.

User Action: Specify the correct virtual device number or link to the disk.

150E No files of the requested type were found

Explanation: There are no VPARS files of the requested type in the system. The type corresponds to configuration definitions, concatenation definitions, TPF device table definitions, or TPF format table definitions.

User Action: None.

160W VPARS force close scheduled by *userid*

Explanation: An authorized user has issued a privileged VPARS command to close your VPARS database.

System Action: Your VPARS database is closed.

User Action: Check with your system programming group.

163E Database disk *vdev* is in use by VPARS database with key *dbkey*

Explanation: The disk you have linked at device number *vdev* is already in use as a database disk by another user. You cannot open the database using this disk. For VPADD, you cannot add this disk.

System Action: For VPOPEN, your VPARS database is not opened. For VPADD, the disk is not added.

User Action: Determine why the disk you have linked at *vdev* is being used by another database, or detach that disk and retry the open or add.

164E VPADD failed

Explanation: This message is issued after an I/O error message for a disk you are trying to add to your database.

System Action: The disk is not added.

User Action: Make sure the disk you are trying to add has been formatted with the VPFMT command.

165E Disk *vdev* is in use by VPARS and cannot be redefined

Explanation: You cannot redefine (change the virtual address of) a minidisk that is being used by VPARS, either as a database disk, a TPF disk whose I/O is being intercepted by VPARS, or a pool minidisk.

System Action: The redefine does not take place.

User Action: If you need to change the address of a disk: For an intercepted TPF base disk, you can use VPSET INTERCEPT OFF, redefine the disk address, and use VPSET INTERCEPT ON. For a VPARS database minidisk, you must close your database, detach the disk, relink it at the correct address, and open the database.

166E TPF record *cchhr* does not match format table *vdev vdev*

Explanation: This message is only issued in a NOBASE system. The record number in file address *cchhr* is not valid based on the format table for this device.

System Action: VPARS returns a unit check condition with No Record Found indicated in the sense bytes. The TPF application may also issue a similar message.

User Action: Make sure the format table is correct, and determine why the application is specifying an invalid file record number. For the TPF commands ZAFIL and ZDFIL, specify a valid file address.

167E Disk *vdev* is part of an open VPARS database with key *dbkey*

Explanation: You cannot detach a disk that is being used by a VPARS database. VPARS needs to perform I/O to the database disk.

System Action: The disk is not detached.

User Action: If you need to detach the disk, you must close the database first.

168E Disk *vdev* is not linked by *userid*

Explanation: For VPADD, one of the users who has the loosely coupled database open does not have the disk at *vdev* linked.

System Action: The disk is not added to the database.

User Action: In order to add a disk to a loosely coupled database, all users who are coupled to the database must first link it read/write.

169E VPARS pool disk *vdev* cannot be used in a coupled database

Explanation: All users in a loosely coupled database must link to a disk before it can be added to a database. Therefore, pool disks are not allowed to be part of a loosely coupled database.

System Action: For VPADD, the pool disk is not added to the database. For VPOPEN, the database is not opened.

User Action: If you need to add a disk to a loosely coupled database, all users must link to a non-pool disk (read/write). Then, one user can add that disk to the database. For VPOPEN, you cannot use the COUPLED option if you have any pool disks linked in your VPARS database minidisk range.

170E VPARS database disk *vdev* is linked read/only by *userid*

Explanation: For VPADD, one of the users who has the loosely coupled database open does not have the disk at *vdev* linked read/write.

System Action: The disk is not added to the database.

User Action: In order to add a disk to a loosely coupled database, all users who are coupled to the database must first link it read/write.

171E Device at address *vdev* is not a disk for user *userid*

Explanation: For VPADD, one of the users who has the loosely coupled database open has a device at address *vdev* that is not a minidisk.

System Action: The disk is not added to the database.

User Action: In order to add a disk to a loosely coupled database, all users who are coupled to the database must first link it read/write.

172E Disk *vdev* for user *userid* is not the correct disk

Explanation: For VPADD, one of the users who has the loosely coupled database open has a device at address *vdev*. That is not the same minidisk as the minidisk you are trying to add. The minidisk is the same if it is on the same real DASD, and has the same starting and ending cylinder extents. In other words, the minidisks should be from the same MDISK definition in the CP directory.

System Action: The disk is not added to the database.

User Action: In order to add a disk to a loosely coupled database, all users who are coupled to the database must first link it read/write.

173E Disk *vdev* for database with key *dbkey* overlaps your disk *vdev*

Explanation: A disk that you were trying to use in a database (in VPOPEN or VPADD) is overlapped by a disk that is part of another user's database.

System Action: For VPADD, the disk is not added. For VPOPEN, the database is not opened.

User Action: Make sure all MDISK statements or LINK commands are correct, and determine why there is an overlap in minidisk definitions.

174E Disks cannot be removed from a shared database

Explanation: You cannot remove a disk from a database that is being used by more than one user.

System Action: The disk is not removed.

User Action: If you need to remove the disk, all users on that database should close it and detach the disk, and then reopen. Alternatively, all users but one can close the database, and that user can remove the disk.

175E Disk *vdev* does not match database disk.

Explanation: All users who are using a loosely coupled database must have links to all the database disks. Your disk at address *vdev* is not the same disk that other users have at that address.

System Action: The database is not opened.

User Action: Make sure all MDISK statements or LINK commands are correct, and determine why the minidisk is not the correct one.

176E Unable to acquire lock to join database with key *dbkey*

Explanation: VPARS could not acquire an internal processing lock to allow you to join a loosely coupled database as an additional user. The VPOPEN command processor tries to obtain this lock for two minutes. If the lock cannot be obtained in this period of time, this error message is issued and the VPOPEN terminates.

System Action: The database is not opened.

User Action: Check with your system programming group.

System Programmer Action: Notify Virtual Software Systems of the error.

177E Locks are not available for requested function, try request later

Explanation: VPARS could not acquire an internal processing lock that is required to perform the requested command. If the required lock cannot be obtained in a few seconds to two minutes, rather than waiting indefinitely, this message is issued and the command terminates.

System Action: The command is not completed.

User Action: Check with your system programming group.

System Programmer Action: Notify Virtual Software Systems of the error.

178E Request is invalid for a loosely coupled database

Explanation: Certain commands are not allowed for a loosely coupled database.

System Action: The command is not performed.

User Action: If you need to perform the function, close the database and reopen it without the COUPLED option.

179E Sense=*sense*

Explanation: This message displays 32 bytes of sense data to accompany other error messages.

User Action: See the explanation of the other message(s) issued along with the sense data.

180E Configuration *cfgname* appears twice in open request

Explanation: The database represented by the configuration named *cfgname* appears in the open request more than once. The "open request" is all of the configurations that appear in the named (or defaulted) concatenation, plus the named or defaulted configuration. Even if the configuration name itself is not repeated, two different configurations in the concatenation may specify the same base minidisk.

System Action: The open does not complete.

User Action: Make sure that the default (or specified) configuration name does not appear in the concatenation definition. A concatenation definition should only contain the configuration names that are to be opened read/only in addition to the read/write configuration. Also, make sure that different configurations named in the concatenation definition do not specify the same base minidisk number.

If you want all the databases named in the configuration opened read/only, you may need to specify "noconfig" on the open.

185E IORCPA=*address* CCW=*ccw* REQADR=*address* RETADR=*address* Caller=*name* at *address*

Explanation: This message is part of the I/O error message.

IORCPA=*address*

The host address of the real channel program.

CCW=*ccw*

The failing CCW.

REQADR=*address*

The host address of the I/O request call to HCPVQ6.

RETAADR=*address*

The host address requested for the I/O complete return.

Caller=*name*

The name of the module that requested the I/O

at *address*

The host address of the module that requested the I/O

186E Shared data base disk addresses for level *nn* conflict

Explanation: During open, your disk addresses at the specified level did not match the addresses that other users were using for the database at that level.

nn The level of the concatenation with the address mismatch.

User Action: Check with your system programming group.

System Programmer Action: If you need further clarification, contact Virtual Software Systems.

187E Your disks at *nn1* must be linked at *nn2* to join this database.

Explanation: During open, your disk addresses did not match the addresses the database was open with.

nn1 The device number of your base disk.

nn2 The device number of the base disk the database is open with.

User Action: Check with your system programming group.

System Programmer Action: If you need further clarification, contact Virtual Software Systems.

188E size of VPARS minidisk *vdev* is [larger | smaller] than size when minidisk was formatted

Explanation: The disk at address *vdev* was one size when it was formatted, and it is a different size now. The disk may have been redefined in the CP directory, or it may have been overlaid (such as with DDR) with data from another database disk.

System Action: The database is not opened.

User Action: Format the database disk at address *vdev*

189E Clear is not restricted, RESTRICTED keyword option is invalid.

Explanation: The RESTRICTED keyword option is not allowed on VPCLEAR, VPIPL, or VPOPEN if the database to be cleared is not restricted.

190E Clear is restricted, RESTRICTED keyword is required to clear.

Explanation: The RESTRICTED keyword is required on VPCLEAR, VPIPL, and VPOPEN if the database to be cleared is restricted.

191E Restricted name *cfgname1* does not match open configuration *cfgname2*

Explanation: The restricted name entered on the VPCLEAR, VPIPL, or VPOPEN command does not match the configuration name of the data base to be cleared.

192E RESTRICTED keyword option was specified without a clear option

Explanation: The RESTRICTED keyword cannot be specified on VPIPL or VPOPEN if a CLEAR or CHKPOINT option is not also specified.

193W You have linked your maximum of *nnn* pool disks

Explanation: A limit of *nnn* pool disks has been set for any one user to link, and you have linked that many pool disks.

System Action: You are not allowed to link any more pool disks.

User Action: Check with your system programming group.

195E Concatenated configuration does not match existing loosely coupled configuration

Explanation: You are trying to open a multi-level database with the read/write level loosely coupled with other users. However, the other users who are coupled to the read/write database have a different set of read/only databases. This mismatch could cause incorrect records to be returned to the TPF application, and your installation has chosen to restrict a VPOPEN with this kind of mismatch.

User Action: If you need to open the loosely coupled database, you must use the same concatenation as the other users who are already coupled to the read/write level.

System Programmer Action: Whether or not this mismatch in the read/only levels will prevent a VPOPEN is defined in the configuration definition for the read/write level.

196E Base VPARS minidisk for configuration you opened with changed, link failed

Explanation: VPLINK is trying to use information from the configuration you used to open your database to determine where to link the new disk (the "link-as" address) and the number of disks in the database range. The definition of the configuration that you used to open your database has been changed since your database was opened. That configuration now represents a different set of database minidisks, and the allowable number of disks in the range may have also changed. The VPLINK command can't use information from a configuration that has been redefined.

System Action: The link is not performed.

User Action: If you need additional database minidisks, you will have to close and reopen your database. However, since someone changed the starting disk address in the configuration you were using, you will also have to relink your database disks at the new addresses. Contact your system programming group.

200E *vdev* ,not added, as it would exceed the maximum allowed (256) Mdisks

Explanation: When adding minidisks to a database, the maximum number allowed has been reached. *vdev* is the first address exceeding the maximum.

System Action: The minidisk is not added.

User Action: You cannot add anymore database minidisks. If you need more space, you need to enlarge existing minidisks.

201W **Reset function suspended, waiting on virtual device at address *vdev* for user *userid***

Explanation: VPARS is in the process of closing a database, but some I/O are still outstanding for the virtual device *vdev*

System Action: The close function will stop, and the message will be displayed every 5mn until the outstanding I/O operation finishes.

User Action: Contact your your system programming group.

System Programmer Action: Determine why the I/O is pending at the device.

202E **RVP component ID unavailable. Return code *nn* from macro HCPXSERV**

Explanation: VPARS component ID block became unavailable.

User Action: Contact your your system programming group.

System Action: Most likely a VPARS internal problem. Note the complete error message number and text of the message and contact Virtual Software Systems.

203E **VSSI user parm disk at address *vdev* owned by userid *userid* is not currently accessed**

Explanation: The user issuing the command or function has a personal parm disk. At this moment, the parm disk at address *vdev* which is owned by *userid* is not CP accessed.

User Action: Contact your your system programming group, or issue the command VPACC *fmode*

System Action: Issue the command CPACC *userid vdev c RR* to make the minidisk available to VPARS.

204E **There is no VPARS private parameter disk for user *userid*. The command is cancelled.**

Explanation: The command VPACC or VPREL has been issued by a user who does not have a private parameter disk.

205E **VPARS private parameter disk *vdev* owned by *userid* is already accessed as *fmode***

Explanation: The command VPACC *fmode* has been issued, but the user's private parm disk is already accessed.

User Action: If the disk really needs to be reaccessed, issue the VPREL command first.

206E **The requested file mode is not available. Disk *vdev* owned by *userid* is currently using file mode *fmode***

Explanation: The command VPACC *fmode* has been issued, but the filemode specified is already allocated to a disk other than the user's private parameter disk

User Action: Re-issue the VPACC with a different *fmode*

999S **Invalid message address, return offset *nnn***

Explanation: A message was to be issued, but the address of the message was invalid or overlaid. The message processing module issues this message to report this condition.

System Programmer Action: Please Note the complete error message number and text of the message and contact Virtual Software Systems.

Chapter 18. VPARS CMS Messages

This section lists the messages issued by the VPARS CMS utilities. The message number, explanation, and user actions are shown. The full message number is DMSxxxnnnt, where:

xxx is the first three characters of the name of the module that issued the message, such as VPB for VPBKUP.

nnnn

is the message number. VPARS CMS messages start at message number 2000.

t

is the message type:

- I for informational messages
- W for warnings
- E for errors
- S for severe errors
- R for messages that require a response (such as YES or NO)
- A for messages that require action (such as mounting a tape)

2000S I/O error on *vdev* CCHHR=*cchhr*, RCODE=*nn*, CPA=*address*, CASC=*ccw*, SENSE=*nnnn*

Explanation: An I/O error occurred during processing of a VPARS database disk.

vdev the disk virtual device number.

cchhr

the cylinder, head, and record of the last seek request.

nn

the diagnose A8 return code.

address

the virtual address of the channel program.

ccw

the CASC data from the I/O request.

nnnn

the first 8 sense bytes returned by diagnose A8.

User Action: Notify your system programming group.

System Programmer Action: The problem may be due to a hardware I/O error on a disk. If the problem persists, note the complete error message number and text of the message and contact Virtual Software Systems.

2001R Format disk on *vdev*? Enter YES or NO.

Explanation: This is a request for verification to format a disk at address *vdev*.

User Action: If the disk should be formatted, enter YES; if not, enter NO.

2002W Error formatting, *vdev* is unuseable

Explanation: The formatting of a database disk at address *vdev* has abnormally terminated. The disk is not usable as a database disk.

User Action: Correct the error in the message preceding this message and reenter the command. If you cannot determine the cause of the error, contact Virtual Software Systems

2003E Subparameter missing or invalid after *xxx* keyword

Explanation: A keyword *xxx* was entered on a command. The required subparameter for the keyword was not entered or is invalid.

User Action: Reenter the command with a valid keyword and subparameter.

2004E Value missing or invalid after *option* option

Explanation: An option was entered on a command. The required value for the option was not entered or is invalid.

User Action: Check the syntax of the command. Reenter the command with a valid value.

2005E Option *option* conflicts with a previous option

Explanation: An option conflicts with another option entered on the same command.

User Action: Check the syntax of the command. Reenter the command with options that do not conflict.

2006E Requested restart cylinder *nn* exceeds minidisk size

Explanation: You used the RESTART option to request that a minidisk format start at cylinder *nn*, but the minidisk you are formatting does not have *nn* cylinders.

User Action: Check the size of the minidisk and reenter the command.

2007E Unexpected operand *xxx*

Explanation: An operand, *xxx*, was specified that is not required or supported by the command.

User Action: Check the syntax of the command and reenter it.

2008E Zero cylinders processed on *vdev*

Explanation: No cylinders were processed by the VPBLDFMT command due to other errors. A previous message should have been issued identifying the error.

System Programmer Action: If you cannot determine the cause of the error, contact Virtual Software Systems.

2009I Building format table for volume *volser* on **vdev *vdev***

Explanation: This is an informational message issued by VPBLDFMT to display the volume serial and the virtual device number of the disk it is processing.

2010I Format table *fname ftype fmode* has been built

Explanation: This is an informational message issued by VPBLDFMT to display the filename of the text file built. The text file can be loaded as a VPARS format table.

2012S I/O error on *vdev* CCHHR=*cchhr*, CASC=*casc*, SENSE=*nnnn*

Explanation: An I/O error occurred during processing of a VPARS database disk.

vdev the disk virtual device number.

cchhr

the cylinder, head, and record of the last seek request.

casc the CASC data from the I/O request.

nnnn

the first 8 sense bytes returned by CP diagnose A8.

User Action: Notify your system programming group.

System Programmer Action: The problem may be due to a hardware I/O error on a disk. If the problem persists, note the complete error message number and text of the message and contact Virtual Software Systems.

2013E Required disk *vdev* is not available.

Explanation: The required disk is not linked or attached to your virtual machine.

User Action: Link or attach the required disk.

2014E *vdev* is not a disk or is an unsupported disk type.

Explanation: The device at address *vdev* is not a disk.

User Action: Link or attach the required disk.

2015E Unknown option *option*

Explanation: An unknown option was entered on a command.

User Action: Reissue the command with valid options.

2016E Required device number was not specified.

Explanation: A command was entered without a required device number.

User Action: Reissue the command with a device number.

2017E Invalid device number was specified *vdev*

Explanation: A command was entered with an invalid device number.

User Action: Reissue the command with a valid device number.

2018E Invalid parameter *xxx*

Explanation: A command was entered with an invalid positional parameter.

User Action: Reissue the command with a valid parameter.

2019E Required file name was not specified.

Explanation: A command was entered without a required file name.

User Action: Reissue the command with the required file name.

2020S I/O error on *vdev* Block=*nnn*, Rcode=*nn*, CPA=*address*, CASC=*cas*, Sense=*nnnn*

Explanation: An I/O error occurred during processing of a VPARS database disk.

vdev the disk virtual device number.

nnn the block number where the error occurred.

nn the return code from CP diagnose A8.

address

the virtual address of the channel program.

cas the CASC data from the I/O request.

nnnn

the first 8 sense bytes returned by CP diagnose A8.

User Action: Notify your system programming group.

System Programmer Action: The problem may be due to a hardware I/O error on a disk. If the problem persists, note the complete error message number and text of the message and contact Virtual Software Systems.

2021E Invalid file mode *mode*

Explanation: An invalid file mode was found in an FS macro ie (FSREAD, FSWRITE). This is an internal error.

User Action: Notify your system programming group

System Programmer Action: note the complete error message number and text of the message and contact Virtual Software Systems.

2022E *vdev* is linked read only.

Explanation: A command was entered that requires write access to a disk. The disk device number specified is linked read-only.

User Action: Link the disk with write access or use a different disk.

2023I Disk *vdev* was not changed.

Explanation: A disk format request was canceled prior to any format operations. If you did not reply NO to the format request, a prior error message was issued that explained the reason for cancellation.

User Action: Correct the error and reissue the format request.

2024I **Formatting disk** *vdev*

Explanation: Formatting of the disk has started.

User Action: None.

2025I ***nn* cylinders formatted on** *vdev*

Explanation: Shows the number of cylinders formatted by the format request.

User Action: None.

2026E **Unknown VPUTIL function &1**

Explanation: The possible VPUTIL functions are DELETE, LIST, MOVE, and PRINT.

User Action: Reissue the VPUTIL command specifying one of the acceptable functions.

2035E *FSmacro* **error** *retcode* **processing file** - *fname ftype fmode*

Explanation: An FS macro (FSREAD, FSWRITE, etc.) returned a nonzero return code processing the specified file. A separate message was issued that explains the return code.

User Action: Correct the error and reissue the command.

2036E **Unknown FS macro command - &1**

Explanation: An unknown FS macro was issued. This is an internal error.

User Action: Notify your system programming group

System Programmer Action: note the complete error message number and text of the message and contact Virtual Software Systems.

2037E **Unknown return code** *retcode* **for** *FSmacro* **macro.**

Explanation: An unknown return code was returned by the specified macro. This is an internal error.

User Action: Notify your system programming group

System Programmer Action: note the complete error message number and text of the message and contact Virtual Software Systems.

2038W Disk *mode* is not accessed or is accessed read-only.

2038W Control and cycle files will not be built.

Explanation: The backup and restore utilities require write access to write the control and cycle files.

User Action: If the disk is accessed read-only intentionally, no action is required. If the disk should have write access, halt the command, reaccess the disk, and reissue the command.

2050E Error validating tape drive *vdev*

Explanation: The CMS utilities validate the tape drive at address *vdev*. A prior message should have been issued identifying the error.

User Action: Correct the tape drive error and rerun the backup or restore.

2051E Tape drive *vdev* is not a virtual drive

Explanation: The VTAPE option was specified for automatic mounting of tapes for a CMS function, but the tape drive at address *vdev* was not a virtual tape drive.

User Action: If you want to use virtual tapes for the restore, detach the device at *vdev* and define a virtual tape drive, or select another address. If you are using real tapes, do not use the VTAPE option on the command.

2052E Uncorrectable error labeling tape on *vdev*

Explanation: A command received an error writing a label to a tape at address *vdev*. An I/O error message should have also been issued.

User Action: Correct the tape drive error if possible, and reenter the command.

2053E Invalid tape function *xxx*

Explanation: If this error occurs, the parameter list used for the WRTAPE or RDTAPE macro may be invalid.

User Action: Notify your system programming group.

System Programmer Action: Note the complete error message number and text of the message and contact Virtual Software Systems.

2054E I/O error tape *vdev*

Explanation: An I/O error occurred during an RDTAPE or WRTAPE function. *vdev* is the virtual tape drive address.

User Action: Use a different tape drive or volume for the function being performed.

2055E Invalid tape drive &1 or drive is not attached

Explanation: The device at *vdev* is not a tape drive, or the internal parameter list used for the WRTAPE or RDTAPE macro is invalid.

User Action: Notify your system programming group.

System Programmer Action: If you cannot determine the cause of the problem, note the complete error message number and text of the message and contact Virtual Software Systems.

2056E Unable to mount virtual tape [*pnnnnn* | SCRATCH] on *vdev*

Explanation: A command was unable to mount a virtual tape on the tape drive at *vdev*. The message will indicate the volume serial of the tape, or the word SCRATCH.

User Action: If you are unable to determine the cause of the error, note the complete error message number and text of the message and contact Virtual Software Systems.

2057E Error *nn* opening [control | cycle | volume] file *fname ftype*

Explanation: An error occurred attempting to open the file named *fname ftype* for a command. *nn* is the error returned by the FSOPEN macro.

User Action: If you cannot determine the cause of the error, note the complete error message number and text of the message and contact Virtual Software Systems.

2059R Continue? - Enter Y for Yes or N for No.

Explanation: This message is used with several non-critical errors, to allow you to make the decision if processing should continue.

User Action: If the current command should continue, enter a Y; if not, enter an N.

2060R Enter next volume serial or 'EOJ'

Explanation: This message is issued at end of volume for a backup tape if the BKTAPE option was used to select the tape for a restore.

User Action: Enter the volume serial of the next tape to process, or enter EOJ if done.

2061R Volume serial entered *volser*. Correct?

2061R Enter Y for Yes or N for No.

Explanation: This message is issued to verify a volume serial, *volser*, that was entered.

User Action: If the volume serial is correct, enter a Y; if not, enter an N.

2062E *volser* wrong length record block *nn*

Explanation: The length of block *nn* read from tape *volser* did not match the requested length.

User Action: Check to ensure the correct tape is mounted.

2063E I/O intercept error VDEV *vdev* HNDIO error code *n*

Explanation: The CMS I/O intercept function HNDIO returned an error indication for the specified VDEV.

User Action: Notify your system programming group.

System Programmer Action: Note the complete error message number and text of the message and contact Virtual Software Systems.

2064E *program* must be run in an XA or ESA guest CMS system

Explanation: The program named *program* that you tried to run cannot execute in a 370-mode virtual machine. You must set your virtual machine to XA or ESA (or XC) mode, IPL CMS, and enter the command again.

User Action: If you need more information, contact Virtual Software Systems

2065E I/O error tape *vdev* block *nn* SCSW *scsw* Sense *sdate*

Explanation: An I/O error occurred processing the *nn* block on the tape drive *vdev*. The Subchannel Status Word and sense data can be user to determine the cause of the I/O error.

User Action: If possible, correct the cause of the I/O error and reissue the command.

2066E Tape drive *vdev* is not attached.

Explanation: A requested or required tape drive is not attached or defined.

User Action: Reissue the request using an available drive or attach or define the drive.

2067E *vdev* is not a tape drive or is an unsupported tape drive type.

Explanation: A requested or required tape drive is not valid.

User Action: Reissue the request using a valid tape drive.

2068E Tape on *vdev* was manually unloaded.

Explanation: During processing, the tape was manually unloaded.

User Action: Determine why the tape was manually unloaded and reissue the command that was processing the tape.

2069E Tape on *vdev* is file protected.

Explanation: A function that writes to a tape determined that the tape was file protected (protected against being written to).

User Action: Obtain write access to the tape and reissue the CMS command.

2100E VPARS database must be [open | closed]

Explanation: The VPARS database must be open or closed, as specified in the message, to execute the requested command.

User Action: Open or close the database. If you close the database, use the NORESET option to prevent your virtual machine from being reset.

2101I Backing up *vdev*

Explanation: This message is issued for each virtual disk processed when a VPARS database is dumped to tape.

User Action: None.

2102I VPARS backup complete, *nn* blocks written, *nn* temporary write errors

Explanation: This message displays the number of blocks written to tape and the number of temporary write errors. A temporary write error is an I/O error that occurred while writing to tape, that was corrected by either retrying the I/O or by performing an 'erase gap' to skip a bad section of the tape.

User Action: If there were any write errors, notify your system programming group that you are receiving I/O errors. The tape drive may need to be cleaned or the tape itself may need to be discarded.

2103E *vdev* is not a VPARS disk

Explanation: The minidisk at address *vdev* is not a VPARS database minidisk.

User Action: Verify that the minidisk at *vdev* is a VPARS database minidisk and that it has not been written over. If you do not have the correct disks linked, relink the correct disks and reenter the command.

2104E Multiple disk sequence error, *vdev* is *nn*, should be *mm*

Explanation: An error has been detected in the minidisk sequence of a VPARS database. *vdev* is the virtual address of the minidisk in error. *nn* is the sequence number that was written to that minidisk when it was last part of a VPARS database. *mm* is the sequence number that should be found on the disk at that address.

User Action: Verify that the minidisk at *vdev* is a VPARS database minidisk and that it has not been written over. If you do not have the correct disks linked, relink the correct disks and reissue the VPOPEN command to open the database, or use the VPOPEN CLEAR command to open and clear the database.

2105E Invalid control record identifier disk *vdev*

Explanation: The VPARS control record read from minidisk *vdev* does not have a valid record identifier.

User Action: Verify that the minidisk at *vdev* is a VPARS database minidisk and that it has not been written over.

2106E Multiple disk control key error disk *vdev*

Explanation: During validation of the minidisks in a VPARS database, minidisk *vdev* was found to have a control key that does not match the key on the first database minidisk. The minidisk at *vdev* may have been reformatted, or used in another VPARS database.

User Action: If the data on the minidisk has been overlaid, the database is no longer usable. It must be opened with the CLEAR option, or the first database minidisk must be reformatted.

2107E VPARS database with key *dbkey* is open for [input | output]

Explanation: The database with the key listed is already open. You cannot perform the requested function while the database is open.

User Action: Use the VPQUERY command to find out who has the database open, or wait until it is no longer open.

2110I Ipl text initialized on *vdev*

Explanation: Ipl text and boot strap records have been written to the requested device.

User Action: None.

2111E Invalid record identifier *id* in file *fname ftype*

Explanation: The requested IPL or bootstrap text file contains an invalid identifier in columns 1-4. Column one must contain hex 02 (X'02'). Valid identifiers are ESD, TXT, RLD, SYM, and END. An * in column 2 is also valid.

User Action: Examine the text files to identify the invalid record.

2112E Error reading VOL1 label from *vdev*

Explanation: An I/O error occurred attempting to read a disk volume label from minidisk *vdev*. Message 2000S I/O error will follow this message.

User Action: Correct the cause of the I/O error and reissue the command.

2113E Error writing track zero on *vdev*

Explanation: An I/O error occurred attempting to write the IPL records and label to minidisk *vdev*. Message 2000S (I/O error) will follow this message.

User Action: Correct the cause of the I/O error and reissue the command.

2114E Ipl file *fname ftype* length *nnnn* exceeds maximum of *nnnn*.

Explanation: The specified file exceeds the maximum length for the type of file on the requested disk.

User Action: If the specified file is correct, note the complete error message number and text of the message and contact Virtual Software Systems.

2116E Ipl [record 2 | text] text length ESD entry not found

Explanation: An ESD type 00 or 04 entry was not found in the ESD record for the specified file, or there is no ESD record in the file.

User Action: If you cannot determine the cause of the error, note the complete error message number and text of the message and contact Virtual Software Systems.

2118E Ipl bootstrap ID is not O/S or TPF

Explanation: The IPL bootstrap text file contains an invalid header.

User Action: Review the VPIPLWR command description and bootstrap sample assemble assemble files. If you cannot determine the cause of the error, note the complete error message number and text of the message and contact Virtual Software Systems.

2119E Ipl bootstrap header contains invalid adcon count

Explanation: The IPL bootstrap text file contains an invalid header.

User Action: Review the VPIPLWR command description and bootstrap sample assemble assemble files. If you cannot determine the cause of the error, note the complete error message number and text of the message and contact Virtual Software Systems.

2120E TPF read CCW data count *n* is less than IPL text length *n*

Explanation: The data length for all CCWs if the TPF bootstrap text file is less than the length from the ESD record in the IPL text file.

User Action: Check the TPF bootstrap assembly file for the correct number of CCWs and the correct length in each CCW. Check the IPL text assembly listing for the length of the IPL text. If you cannot determine the cause of the error, note the complete error message number and text of the message and contact Virtual Software Systems.

2121E Read CCW adcon does not point to read CCWs

Explanation: The first byte pointed to by the read CCW adcon is not a read CCW opcode X'06'.

User Action: Check the bootstrap assembly and listing files to determine the error. If you cannot determine the cause of the error, note the complete error message number and text of the message and contact Virtual Software Systems.

2130I Restoring vdev

Explanation: This message is issued for each VPARS database disk to which active blocks are restored.

2131I VPARS restore complete, nn blocks restored

Explanation: The requested VPARS database restore is complete, and *nn* database blocks were restored.

2132E Insufficient blocks nn on vdev to restore nn blocks

Explanation: Your VPARS database disk *vdev* is not large enough to hold the data on the tape you are trying to restore.

User Action: Increase the size of the disks in your VPARS database and reissue the VPREST command. Each database disk must be large enough to hold the data that was on the disk during the backup.

2133E volser is not a VPARS [backup | unload] tape

Explanation: The first record on the tape mounted is not a VPARS backup or unload header. A tape created by VPBKUP has a backup header and can be restored with VPREST, and a tape created by VPUNLD has an unload header and can be restored with VPUNLD. *volser* is the volume serial of the tape.

User Action: Verify that the correct volume is mounted and has not been overwritten. If the tape is valid, note the complete error message number and text of the message and contact Virtual Software Systems.

2134E Error nn mounting [backup | unload] tape

Explanation: Return code *nn* was returned from VSLABSL attempting to mount a tape. Return code 3 is used when you respond T to Terminate a program.

User Action: A prior error message was issued by VSLABSL. See the user action for that message.

2135E VPARS control record error on tape *volser*

Explanation: The VPARS control record read from the tape is not valid. *volser* is the tape volume serial.

User Action: Verify that the correct volume is mounted and has not been overwritten. If the tape is valid, note the complete error message number and text of the message and contact Virtual Software Systems.

2136E [Backup | Unload] tape block *nn* is out of sequence on *vdev*

Explanation: The blocks on the tape are out of sequence. An I/O error may have occurred during reading or writing the tape. In attempting recovery, a block was lost.

User Action: Retry the load or restore. If the error persists, note the complete error message number and text of the message and contact Virtual Software Systems.

2137E Tape record *nn* is not a valid VPARS [backup | unload] record

Explanation: The header on the tape was incorrect. An I/O error may have occurred during reading or writing the tape.

User Action: Examine the tape to determine if it has been overwritten. If it has not, note the complete error message number and text of the message and contact Virtual Software Systems.

2138I [Backup | Restore] tape created on *date* at *time*

Explanation: The backup tape being restored was created on the specified date and time.

2139R Restore VPARS database? - Enter YES or NO.

Explanation: A VPARS restore command has been entered.

User Action: If the database should be restored, enter YES; if not, enter NO.

2140E Insufficient disks for restore, *nn* available

Explanation: A VPARS restore command has been entered, but you do not have enough disks to perform the restore. This message is followed by messages 2141 and 2142 to indicate how much disk space you need.

User Action: See the following messages.

2141I *n* disks are required for a restore of this VPARS backup tape

Explanation: This message indicates the number of disks that you must have linked in your VPARS database address range to restore the backup tape.

User Action: Ensure that you have the required number and size of disks linked and retry the restore.

2142I Disk *vdev* must have at least *nn* blocks

Explanation: This message shows the required size of each disks in your VPARS database, in 4096-byte blocks, to restore the backup tape.

User Action: Ensure that you have the required number and size of disks in your VPARS database address range and retry the restore.

2143S The backup cycle file did not include all tape volumes.

2143S *nn* blocks were not restored. The VPARS database is unusable.

Explanation: There was an error in the backup cycle file, or the file has been modified after it was created. Some of the blocks in the VPARS database were not restored, and the database will not be valid.

User Action: Note the complete error message number and text of the message and contact Virtual Software Systems.

2144R Load TPF records? - Enter YES or NO.

Explanation: This message confirms your request to load TPF records. If you have an open VPARS database, the records will be written to it. If you do not, VPLOAD will attempt to write the records to the TPF base system disks.

User Action: If the VPLOAD should proceed, enter YES; otherwise, enter NO.

2145I *nn* records [loaded | unloaded]

Explanation: This message shows the number of records that were unloaded by VPUNLD, or loaded by VPLOAD.

2150W TPF dasd *vdev* is not available for processing

Explanation: A disk at the specified device address was not found, or an I/O error occurred when trying to read the disk.

User Action: Processing continues for other TPF devices. If you are not receiving hardware I/O errors on the disk, note the complete error message number and text of the message and contact Virtual Software Systems.

2152I *nn* record(s) processed for &2

Explanation: Several VPARS commands issue this message to show the progress in processing.

User Action: None.

2153S Error in VPARS address compute

Explanation: The relative block number was higher than the number of blocks available on the VPARS database.

User Action: Notify Virtual Software Systems

2154E Requested start address is greater than end address

Explanation: The ending address of the range must be larger than or equal to the starting address.

User Action: Enter the command with a valid address range.

2155E Delete error in directory index index block

Explanation: An error was found in the VPARS directory structure that would have caused the highest level index block to be deleted.

User Action: Note the complete error message number and text of the message and contact Virtual Software Systems.

2156E Record identifier check *xx vdev=vdev, cchhr=cchhr*

Explanation: A record read from VPARS database disk *vdev* did not have the correct record identifier. *xx* is the correct record identifier. *cchhr* is the VDEV address of the VPARS database disk.

User Action: Note the complete error message number and text of the message and contact Virtual Software Systems.

2158I *nn* TPF records on VPARS database *nn* blocks in use

Explanation: *nn* records were found on the VPARS database *nn* VPARS database blocks are in use.

2159I *nn* TPF records on unloaded database *nn* blocks were in use

Explanation: Shows the number of TPF records unloaded and the maximum number of VPARS database blocks required to load the VPUNLD tape.

2160W Tape record counts are not available.

Explanation: A VPLOAD tape map was requested and the tape that was mounted was created prior to the availability of the map option.

2253E Block sequence error on backup tape

Explanation: During a restore, the tape blocks on the backup tape were found to be out of order.

User Action: Make sure the restore tapes were not mounted out of order.

2254E Block read *nn* should be *mm*

Explanation: This message is issued with message 2253E to identify the block in error.

User Action: See message number 2253E.

2300R Enter 'M' to Remount or 'T' to Terminate

Explanation: This message is issued after a prior message, such as an unlabeled tape being mounted when a scratch was requested.

User Action: Enter M to unload the tape and mount a different scratch tape. Enter T to terminate the program.

2301A Mount *vdev* SL [*volser* | SCRATCH]

Explanation: This message asks for a tape to be mounted for a VPBKUP, VPUNLD, VPREST, or VPLOAD. *vdev* is the virtual device number (address) of the drive on which a tape should be mounted. *volser* is the volume serial of the tape to be mounted, or SCRATCH for a scratch tape.

User Action: Mount the requested tape, or a scratch tape, on the requested drive.

2302E Tape on *vdev* is not standard labeled

2302R Enter 'M' to Remount or 'T' to Terminate

Explanation: The first record on the tape mounted on *vdev* is not a VOL1 label or is not an 80-character record.

User Action: Enter M to unload the tape; you will be prompted to mount the requested tape again. Enter T to terminate the program.

2303E Tape on *vdev* is *volser1* not *volser2*

2303R Enter 'M' to Remount or 'T' to Terminate

Explanation: The wrong tape was mounted on the requested tape drive at address *vdev*. *volser1* is the internal volume serial of the volume mounted. *volser2* is the volume serial that was requested.

User Action: Enter M to unload the tape; you will be prompted to mount the requested tape again. Enter T to terminate the program.

2304E I/O error *n* processing [HDR1 | EOF1] label on *vdev*

2304R Continue? - Enter Y or N.

Explanation: Return code *n* was received for an I/O operation to read a HDR1 or EOF1 label from the tape on drive *vdev*.

User Action: If you want to continue the program, enter Y; if not, enter N.

2305E Dsname on tape *vdev dsname*, does not match requested *dsname*

2305R Enter 'M' to Remount, 'U' to Use the tape or 'T' to Terminate

Explanation: The dataset name on the requested tape mounted on *vdev* does not match the requested dataset name. If the correct tape is mounted, it may have been overwritten.

User Action: Enter M if you want a different tape mounted. Enter U to use the mounted tape. Enter T to unload the tape and terminate the job.

2306I Tape on *vdev* is SL *volser* **dsname=dsname**

Explanation: This message is issued when a tape is mounted for input or output, and the label has been read. *vdev* is the tape drive address. *volser* is the volume serial of the tape. *dsname* is the data set name on the tape.

2307W Invalid SL trailer for tape *volser* on *vdev*, block count check suppressed.

Explanation: This message is issued when the block count check has been suppressed and the count of blocks does not match the trailer count.

User Action: None.

2308W Block count *nn* on tape *vdev* does not match blocks read *mm*

2308R Continue? - Enter Y for Yes or N for No.

Explanation: There is a difference between the block count in the EOF1 label, *nn*, and the blocks read by the program, *mm*, for the tape mounted on *vdev*.

User Action: Enter Y to ignore the error and continue with the next volume. Enter N to terminate the program.

2309A Keep *vdev* SL *volser* **dsname=dsname**

Explanation: This message is issued when a tape is unloaded from tape drive *vdev*.

2310A Mount *vdev* SL scratch

Explanation: This is a request for a standard labeled scratch tape to be mounted on *vdev*.

User Action: Mount a standard labeled scratch tape on the requested tape drive.

2311R Enter 6 character volume serial or 'UNLOAD' for tape on *vdev*

Explanation: An unlabeled tape was mounted on *vdev*.

User Action: Enter a 6 character volume serial to label the tape, or UNLOAD to remove the tape and mount another.

2312I Volume serial entered was *volser*

2312R Correct? - Enter Y for Yes or N for No.

Explanation: This message is issued to verify the volume serial that you entered.

User Action: If the volume serial is correct, enter Y; if not, enter N.

2313W Volume *volser*, **dsname=***dsname*, **on** *vdev*, **expires** *date*

2313R Enter 'M' for Remount or 'U' to Use the tape.

Explanation: The tape that is mounted on *vdev* has not expired.

User Action: Enter M to unload the tape and prompt for another scratch tape to be mounted, or U to overwrite the unexpired tape.

2315E Logical error processing [HDR1 | EOF1] label on *vdev*

2315R Continue? - Enter Y for Yes or N for No.

Explanation: An error occurred reading a HDR1 or EOF1 label from the tape on drive *vdev*.

User Action: If you want to try to run the restore with the tape volume mounted, enter Y; if not, enter N.

2420E Invalid tape record identifier *id*, tape block *nn*

Explanation: The tape record identifier does not match the expected identifier for the command being executed.

User Action: Note the complete error message number and text of the message and contact Virtual Software Systems.

2421E Record was not found in *fname ftype* file for TPF module *module*

Explanation: The TPF disk module to be restored was not found in the module configuration file.

User Action: Add the module entry to the configuration file.

- 2422E Invalid parameter *parm* in *fname ftype* file**
Explanation: The specified parameter is invalid in the file.
User Action: Review the file specifications and correct the parameter.
- 2423I *nn* records processed from tape *volser* for module *module* on *vdev***
Explanation: Shows the number of TPF records read from the tape.
User Action: None.
- 2424I *nn* records restored, *nn* zero records were not restored**
Explanation: Shows the number of TPF records restored from the tape and the number of records containing all zeros that were not restored.
User Action: None.
- 2425I *nn* Long term pool records were restored**
Explanation: Shows the number of long term pool records that were restored.
User Action: None.
- 2426I Restoring module *module* tape *nn* to *vdev volser* tape *volser volser***
Explanation: A restore of a TPF capture tape has been started.
User Action: None.
- 2427I TPF system date and time from capture tape *date time***
Explanation: Shows the date and time the TPF capture was run.
User Action: None.
- 2428I Start CCHH=*cchh*, End CCHH=*cchh***
Explanation: Shows the starting and ending cylinder and head numbers if a range was specified for the restore.
User Action: None.

2440I *fname* **VPBXRfmt** *fmode* **has been renamed to** *fname* **VPBXSfmt** *fmode*

Explanation: The previous format file has been saved with a file type of VPBXSfmt.

User Action: None.

2441E **Return code** *nn*, **renaming** *fname1* *fctype1* *fmode1* **to** *fname2* *fctype2* *fmode2*

Explanation: The previous file *fname1* *fmode1* *fmode2* could not be renamed to the "save" filename *fmode2* *fctype2* *fmode2*.

User Action: Ensure that you have read/write access to your *fmode1* disk.

2443E **Possible bad OPMAAA - check SYCON MACRO**

Explanation: The message indicates one of the following errors.

- There are zero pool intervals.
- No matching RCC intervals were found.
- Only one base SON/FARF slot.
- Not enough record slots for all record sizes.

User Action: Check the SYCON macro for errors. Note the complete error message number and text of the message and contact Virtual Software Systems.

2445E **More OPMAAA entries for RCC than can be processed**

2445E **Check OPMAAA in SYCON MACRO**

Explanation: A maximum of 42 entries are allowed.

User Action: Check your SYCON macro. Note the complete error message number and text of the message and contact Virtual Software Systems.

2449I **Creation of new pool format table** *fname* **VPBXRfmt** *fmode* **is complete**

Explanation: The new format table can be used for VPBXREST processing.

User Action: None.

2450E First noncomment card in range exclude file *fname* *ftype* is not HEX or DEC

Explanation: The first control card in the range exclude file must identify the type of numbers used to specify the ranges.

User Action: Correct the range exclude file.

2451E Start of range *cchh* is greater than end of range *cchh*

Explanation: The starting cylinder and head of a range is larger than the ending cylinder and head.

User Action: Correct the range exclude file.

2452E Exclude range card has less than the four required parameters.

Explanation: The format of a range exclude record is invalid.

User Action: Correct the range exclude file.

2453E Error converting range cylinder or head number *cchh*

Explanation: The data in a range exclude record contains invalid characters.

User Action: Correct the range exclude file.

2454E Record ID *id* is not two characters

Explanation: The format of a record id exclude record is invalid.

User Action: Correct the record id exclude file.

2455I The following cylinder and head ranges will not be restored

Explanation: This message is followed by message 2456I listing the ranges.

User Action: None.

2456I From CCHH *cchh* to CCHH *cchh*

Explanation: This message shows a range of tracks that will not be restored.

User Action: None.

2457I TPF records with the following record IDs will not be restored

Explanation: This message is followed by message 2458I listing excluded record ids.

User Action: None.

2458I *id id id*

Explanation: This message lists the IDs of records that will not be restored.

User Action: None.

Chapter 19. VPARS ABEND codes.

This section lists the VPARS abend codes. Abend codes are issued in the form xxxnnn, where:

xxx identifies the module that issued the abend, such as PRC for module RVPPRC

nnn is the abend number.

An abend of PRC002 is issued by the module RVPPRC and has a code of 002. All VPARS generated abends are SOFT abends but for CON001 which causes a HARD ABEND.

PRC002 Module not in VPOPEN or VPINIT

Explanation: This module is only called during a VPOPEN or VPINIT.

PRC003 Uword mismatch between RVPSYS and RVPPRC

Explanation: The pointer passed from RVPSYS does not exist in RVPPRC

PRC008 Return code greater than 4 when allocating the RVP CMPID block

Explanation: A return code greater than 4 usually indicates that an invalid parameter was passed.

RST002 Missing RVP (VPARS) component ID block

Explanation: The RVP CMPID was expected to be present, but the RVPRST module could not find it

CCW001 Missing VPARS control block.

Explanation: The anchor address for the VPARS control blocks cannot be found.

CON001 HARD ABEND. Missing VDEV for VPMSKBBK

Explanation: When requesting the address of the control block associated with a virtual device, CP told VPARS that the virtual device did not exist.

CON005 Missing Pool disk control block.

Explanation: When handling a request dealing with a pooled Mdisk, the Pool disk control block was not found.

RCC001 Timer request block active.

Explanation: A Timer Request Block ready for scheduling is found to be active.

RCC002 Pool disk is not linked.

Explanation: During close processing, a VPARS pool Mdisk is found not to be linked by the user issuing the close.

SV2001 Missing Mdisk control block.

Explanation: The address of the control block associated with a given Mdisk cannot be found.

SV2002 Database is not close when detaching a Mdisk.

Explanation: During a detach process, a mdisk is found to be active.

SV2003 Missing Pooldisk control block.

Explanation: When handling a request dealing with a pooled Mdisk, the Pool disk control block was not found.

SV2004 VPARS IORBK is active, but the VPARS database is closed.

SV2005 Missing Pooldisk control block.

Explanation: When handling a request dealing with a pooled Mdisk, the Pool disk control block was not found.

DBM002TPF record length is greater than 4096 bytes.

DBM003Incorrect record header.

DBM004Invalid record header.

DBM005Directory/Index index record not found.

DBM006Invalid directory record.

CLR001 A Timer Request Block already exists.

IOR001 Invalid address upon entry to IORUL.

Explanation: The address of the buffer to unlock, contained in R1, is negative.

IOR002 VPARS IO Task Block is already queued.

Explanation: A request to queue an already queued I/O task block has been received.

IOR004 VPARS IO Task Block is no longer queued.

Explanation: A request to dequeue an I/O task block has been received, but the I/O task block is not queued.

IOR005 IO Task Block is on the wrong queue.

Explanation: The I/O Task Block is found on a different queue than the one specified in the request.

IOR006 Invalid header in a Record Directory block.

Explanation: Before scheduling a write operation for a Record Directory block, the block header is checked and was found to be invalid.(ie: not RD)

Chapter 20. VSSI CP Messages

This section lists the VSSI CP messages with explanations, and user actions. Informational messages are usually issued without the message number. The full message number is RVSxxxnnnt, where:

xxx identifies the module that issued the message, such as CMD for module RVSCMD.

nnn is the message "number".

t is the message type:

- I for informational messages
- W for warnings
- E for errors
- S for severe errors
- R for messages that require a response (such as YES or NO)

A typical message number is RVSCMD020E. This message was issued by VSSI module RVSCMD, and is message number 020E.

001I VSSI disk at address *vdev* owned by userid *userid* is not currently accessed

Explanation: The VSSI parm disk is defined, but is not currently CP accessed.

User Action: Issue the CPACC command.

002I There was no definition for the VSSI parm disk in the configuration file used at IPL

Explanation: The VSSI statement VSI_Disk was not in the SYSTEM CONFIG file used to IPL.

User Action: Issue the VSSet VSIDisk command to define the VSSI parm disk.

003E An empty RVS component ID block was found. Contact Virtual Software Systems.

Explanation: The RVS CMPIDBK has been, unexpectedly, reset to X'0's

User Action: Contact VSSI for help.

- 004I** VSSI disk at address *vdev* owned by userid *userid* and accessed as *c*
Explanation: The VSSI parm disk is available.
- 006I** VSSI parm disk set to *vdev*, owned by *userid*
Explanation: The current VSSI parm disk definition.
- 007I** VSSI parm disk was *vdev* owned by userid *userid*
Explanation: The VSSI parm disk definition was changed. This message displays the old values.
- 008I** Specify REPlace to overwrite existing definition.
Explanation: The option (REPlace) is required when changing an existing definition.
User Action: Re-issue the command with the REPlace option.
- 009E** VSSI component ID block not found. Cannot open *fname* file.
Explanation: The RVS CMPIDBK was not found. This could happen if, during IPL, a VSI_Disk statement was not found in the VSSI config file.
User Action: You can create the RVS CMPIDBK with the command VSSet VSIDisk *userid vdev*. You should also verify that a VSI_Disk statement is present in the VSSI config file.
- 010E** Invalid argument *n* passed to RVSCFG.
Explanation: The value *n* is invalid. This is an internal error.
User Action: Contact Virtual Software Systems.
- 011E** File *fname ftype fmode* not found.
Explanation: The specified file was not found.
User Action: Verify that the file in question does reside on the VSSI parm disk, or (if VPARS) on the (optional) configuration disk(s).

- 012E Error encountered while attempting to read records from *fname ftype fmode* record *n***
- Explanation:** The specified file exists, but cannot be read starting at record *n*
- User Action:** Verify that the file can be read by CP, by issuing the command CPTYPE *fname ftype fmode*
- 013E Statement exceeds record length in *fname ftype fmode* record *n***
- Explanation:** The statement at record *n* exceeds the maximum length of 80 characters.
- User Action:** If the statement requires parameters exceeding 80 characters, split it on multiple lines and mark the continuation, by placing a blank and a comma after the last character of each line.
- 014E Previous syntax error was detected in file *fname ftype fmode* record *n***
- Explanation:** The CP parser has already put out a message explaining the error. This message specifies where the error occurred.
- User Action:** Correct the statement in error.
- 015E Missing continuation. Incomplete record in *fname ftype fmode* record *n***
- Explanation:** An EOF was detected while a record continuation was in progress.
- User Action:** Verify that a comma was not inadvertently left on the last statement.
- 016E Combined length of continued records exceeds 4000 in *fname ftype fmode*, record *n***
- Explanation:** The combined length of a multiple line statement, starting at record *n*, exceeds the maximum of 4000 characters.
- User Action:** Inspect the failing statement for extra commas. No VPARS or VTAPE statement is that long!
- 017E Empty file or no valid statements in *fname ftype fmode* record *n***
- Explanation:** If no parser errors were issued, then the file is either empty or all statements are commented out. Otherwise, all statements are invalid
- User Action:** Review the content of the failing file.

020E Invalid option - *ccc*

Explanation: The option specified is not valid for the command entered.

ccc is the option not recognized.

User Action: Correct the error and reissue the command.

021E Operand missing or invalid - *ccc*

Explanation: The command expects an additional operand which was not specified or is not recognized.

ccc, if specified, is the invalid operand.

User Action: Correct the error and reissue the command.

022I VSSI DATEFORMAT = *ccc*

Explanation: Display of the current VSSI date format.

023E Duplicate option - *ccc*

Explanation: The option was specified more than once in the command.

ccc is the duplicate option.

User Action: Correct the error and reissue the command.

024E Invalid date format specified : *ccc*

Explanation: The date format specified on the VSSet DATEF command is invalid.

ccc is the date format in error.

User Action: Verify the date format entered. Make sure that the delimiter is a slash '/'.

025E Invalid delimiter specified : *c*

Explanation: The delimiter specified, *c*, is invalid.

User Action: Since this command accepts any 1 character as operand, it most likely means that the delimiter entered is longer than 1 character. Correct and reissue the command.

026I This CP is not Year 2000 compliant. The command is cancelled and the date format not changed

Explanation: The command VSSet SYSDate was entered on a system whose Control Program is not Year 2000 compliant. Under this condition, there is no default date format. The command is cancelled.

System Programmer Action: Upgrade to a YEAR2000 compliant CP!

1001I VSSI parm disk at address *vdev* is owned by userid *userid*

Explanation: Definition of the VSSI parm disk as specified on the VSI_Disk initialization statement.

1002I *ccc* configuration statement is duplicated. It will be ignored

Explanation: If an initialization statement is duplicated, only the first instance is considered.

ccc is the duplicated statement.

User Action: The system will initialize with the value specify on the first occurrence of the statement.

Chapter 21. VSSI Abend codes

This section lists the VSSI abend codes generated by the VSSI common code. Abend codes are issued in the form xxxnnn, where:

xxx identifies the module that issued the abend, such as INI for module RVSINI.

nnn is the abend number.

An abend of INI008 is issued by the module RVSINI and has a code of 008. All VSSI generated abends are SOFT abends.

CFG001 Internal error. Filename missing.

Explanation: When requesting the reading of a file, the calling module did not provide a filename.

CFG008 A return code greater than 4 was received when allocating the RVS component ID block.

Explanation: A return code greater than 4, usually means that an invalid parameter was provided to the callee.

INI008 A return code greater than 4 was received when allocating the RVS component ID block.

Explanation: A return code greater than 4, usually means that an invalid parameter was provided to the callee.

PRM001 The RVS component ID block was not found.

Explanation: The RVS CMPID is allocated at initialization time. It should exist for the life of the current IPL.

PRM008 A return code greater than 4 was received when allocating or locking the RVS component ID block.

Explanation: A return code greater than 4, usually means that an invalid parameter was provided to the callee.

This page left intentionally blank

Reader's Comment Form

We welcome your comments on this manual. Is there anything you especially like or dislike about the organization or presentation? Possible topics for comments include clarity, accuracy, completeness, and specific errors and omissions.

If you would like a reply, please include your name and address:

Virtual PARS, release 4.2, printed on 08/02/03.

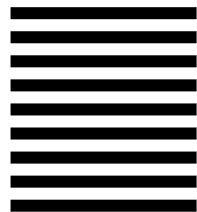
Thank you for your comments. A postage stamp is not necessary if you are mailing this from the U.S.A. You may also fax this page to (770) 781-3210.

Cut along line

Fold and tape



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



BUSINESS REPLY MAIL
FIRST CLASS PERMIT No. 107 Gainesville, GA

POSTAGE WILL BE PAID BY ADDRESSEE

Virtual Software Systems, Inc.
7715 Browns Bridge Road
Gainesville, GA 30506-9867



Fold and tape

Reader's Comment Form

We welcome your comments on this manual. Is there anything you especially like or dislike about the organization or presentation? Possible topics for comments include clarity, accuracy, completeness, and specific errors and omissions.

If you would like a reply, please include your name and address:

Virtual PARS, release 4.2, printed on 08/02/03.

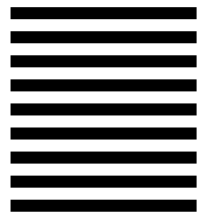
Thank you for your comments. A postage stamp is not necessary if you are mailing this from the U.S.A. You may also fax this page to (770) 781-3210.

Cut along line

Fold and tape



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



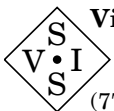
BUSINESS REPLY MAIL
FIRST CLASS PERMIT No. 107 Gainesville, GA

POSTAGE WILL BE PAID BY ADDRESSEE

Virtual Software Systems, Inc.
7715 Browns Bridge Road
Gainesville, GA 30506-9867



Fold and tape



Virtual Software Systems, Inc.

7715 Browns Bridge Road
Gainesville, Georgia 30506
(770) 781-3200 Fax (770) 781-3210