# Using the zCP3000 VM Extract Utility
# CP2KVMXT, Version 2.2h
# September 29th, 2006

**Gretchen Frye**
**frye@us.ibm.com**
**IBM ATS - Capacity Planning Support**

## Table of Contents

# Using the zCP3000 VM Extract Utility

## Preface

This program reads CP monitor data in order to produce a file which is directly useable by ZCP3000 AND CP2000. It is intended for use by customers to extract performance information which will then be used by IBM employees or IBM Business Partners to produce a capacity plan.

CP2KVMXT and its prerequisite packages, Monview and VMARC, may be obtained from the IBM VM Download Packages website at http://www.vm.ibm.com/download/packages/. Use of tools from this site is conditional on the "Agreement for Downloading from the IBM VM Download Library" at http://www.vm.ibm.com/download/license.html Please note that the ZCP3000 modeling tool itself is available only to IBM employees and IBM Business Partners.

IBM employees and IBM Business partners can access additional tools for capacity planning, including ZCP3000, zPCR, and the z/OS extract facility, can be found on the Capacity Planning Support website at Http://w3-1.ibm.com/support/americas/wsc/cpsproducts.html or on the Business Partners website.

## Changes for this version

**Version 2.2h**

- Fix error where the BCUIO and BCUIOV I/O rate is incorrectly set to zero.

**Version 2.2g**

- Support non-cache dasd, handle off/on/off users, new warnings about bad interval lengths & durations.

**Version 2.2f**

- inactive user code fix; system CR fix.

**Version 2.2e**

- The default user class $OTHER has been discontinued. The UCLASS method of grouping users into workloads will continue unchanged, but users that are not included in a UCLASS group will show up as a separate workload instead of being grouped into the $OTHER workload group. Small workloads can be combined into a larger group by selecting and right-clicking on the workloads in zCP3000, but the CPU utilization from very small workloads, such as typical CMS users, will often be "rounded out", and will show up only as a lower capture ratio. If you wish to have the resulting groups remain the same in the EDF file, please see the note at the end of the "Grouping Users Into Workloads" section for actions you can take to make this happen.
- Durations now must be at least 5 minutes long, but in practice, the duration should really be much longer. A duration of 30 or 60 minutes, with a week's worth of data, is recommended.

- Logic changes were made to improve the accuracy of the configuration described in the output EDF file. Further adjustments are sometimes needed in zCP3000, particularly when IFLs, ICFs, zAAPs, or a combination of non-traditional engines are used in the CEC.
- This documentation has been updated to reference zCP3000 instead of CP2000. Data generated by the CP2KVMXT utility will load successfully into either tool.
- A new section in this document explains how to estimate the amount of dasd space that will be needed for a monitor data file.

**Version 2.2d**

- Limited maintenance release sent only to people reporting the problem (some IFL installations).

**Version 2.2c**

- New options to process a subset of the monitor data. You can skip over records before a certain time, and you can limit the processing to a certain number of durations.
- Early warning if it looks like there may not be enough virtual storage to run the utility, based on the number of configured devices in the monitor data.
- More counter wrap problem fixes, for systems with sustained high I/O rates.
- Problem fix for when logical configuration changes during measurement.
- Problem fix for mismatched sample sizes (problem for VM native systems only).

**Version 2.2a**

- **Class B authority is now required to run the utility.**
- counter wrap problem, when SSCH>65,535 per duration
- negative cpu sec when user logoff/logon within duration

**Version 2.2**

- New BCU, PATH, and ACT macros to support DASD and Cache Analysis. **Class B authority is now required to run the utility.**
- fix for counter wrap problem, causing random intervals with CR>1
- warning when workload grouping file not found

**Version 2.1**

- New fields in CEC and SYS macros, to support the configuration of IFL engines
- LCPTYPE added to the input parameters file
- Command line options for input, output, and uclass files can now override the fileids in the parameters file.
- An EDF merge utility is now available upon request to the IBM or Business Partner doing the study, so that EDFs from a single system on multiple days can be merged into one EDF**.**

**Version 1.2.1**

- Make uncompiled BUILDREC the default version.
- Ignore illegal characters in UCLASS file.

- More detail on error messages.

**Version 1.2**
- Support for direct input from the CP Monitor service via IUCV. This greatly reduces the disk space requirement.
- Support for runtime parameter file CP2KVMXT PARMS

**Version 1.1**
- Support for native (non-LPAR) systems
- LPAR management time for all partitions has been summed and moved to a pseudo-partition called PHYSICAL.
- Allow inactive partitions to be included in the EDF.
- General availability to non-IBM employees
- It is no longer necessary to run Monview against the monitor data file to produce a separate, MONVIEW file. The Monview package is still prerequisite because it is called as a routine from CP2KVMXT.
- CP Monitor should also be enabled to collect dasd class I/O samples & events.

## VM Extract Utility Overview:

The ZCP3000 VM extract utility **(**hence referred to as CP2KVMXT**)** generates an Enterprise Data File (EDF) for input into ZCP3000 OR CP2000, which is used by IBMers and Business Partners to generate a capacity plan**.** The utility generates EDF files for VM systems running native as well as those running in a partition; on traditional engines or on IFL engines. The fields and macros generated by CP2KVMXT are documented in an appendix of this document. The utility is designed to be run on the same image that the monitor data was collected on. While it is possible to ship the monitor data elsewhere and run CP2KVMXT against it there, this is discouraged because

1. The sheer volume of data generated by the monitor leads one to try to economize by collecting only a few peak hours. We really want a full week of data, including peak and non-peak time.
2. Information about the channel types is not included in the monitor data, but instead is collected realtime through a set of CP commands at the time CP2KVMXT is run.

## ZCP3000 Overview:

ZCP3000 is a PC-based productivity tool designed to do performance analysis and capacity planning for IBM System z, IBM z Series, and IBM S/390 processors running various SCP/workload environments. The Capacity Planning Support (CPS) team at the Washington Systems Center provides a comprehensive suite of tools designed to assist in capacity planning. Please visit our website at
http://w3.ibm.com/support/americas/wsc/cps.html (for IBM internal)
http://partners.boulder.ibm.com/src/atsmastr.nsf/WebIndex/PRS1762 (for IBM BPs)

or contact your IBM representative for more information about ZCP3000 and other capacity planning tools.

## CP Monitor Requirements:

Capacity planning requires a much longer period of measurement data than is typically used for performance tuning. Running it for one peak hour is not enough. It should be considered an occasional cost of doing business. Preferably, the capacity plan will contain an entire (business) week's worth of data. If your site does not already run the monitor for another service such as ESAMON, VMPRF, or the Performance Toolkit, then you will need to set up this service. For information on the MONDCSS saved segment and running the monitor, please refer to "Chapter 9. Monitoring Performance Using CP Monitor" in SC24-5999 z/VM Performance Guide (see Appendix B for the url of the z/VM Publications library). **The default MONDCSS size is often too small**, especially if there are a lot of devices defined, and the result of this is lost data. If CP2KVMXT is using realtime data, it will shut down if the data is being dumped into the small MONDCSS at a faster rate than CP2KVMXT can process it. See the section entitled
"**Speeding Up CP2KVMXT**", later in this document.

The CP Monitor must be enabled to collect event and sample data for the Processor, Storage, User, and I/O (dasd class) domains. The Monitor sample interval should be fairly short; when running CP2KVMXT with realtime monitor data, it is preferable to use a monitor interval of 1-2 minutes. When saving monitor data to run against later, use an interval of from 2-5 minutes. There are tradeoffs when choosing the interval length:

When running CP2KVMXT with realtime monitor data, shorter intervals increase the overhead of the monitor (because it is gathering data more frequently), and increase the CPU consumption of the userid running CP2KVMXT (because there are more samples to process).

When writing to a monitor data file, in addition to more overhead, short intervals use a tremendous amount of disk space and lengthen the CP2KVMXT processing time.

Longer intervals, on the other hand, run the risk of missing data. Internal counters of frequent events "wrap" and reset to zero once they've reached 65,535. If the number in the counter is less than the last time, CP2KVMXT logically adds 65,535 to get the difference. If it wraps more than once during a monitor interval, that information is lost and your system will look like it has a whole lot less I/O than it really does.

Longer intervals are often incompatible with the requirements of realtime monitors such as IBM's VM/Real Time Monitor or FCON, which need a short interval to gauge the effects of performance tuning.

Monitor overhead can be lessened by monitoring only what you need. CP2KVMXT does not use data from the Scheduler, Seeks, or Application domains, and it uses only dasd class I/O. Here are sample commands to measure just these, sampling at 1 minute intervals:

```
CP MONITOR SAMPLE DISABLE ALL INT 1 MIN
CP MONITOR SAMPLE ENABLE PROC
CP MONITOR SAMPLE ENABLE STOR
CP MONITOR SAMPLE ENABLE USER ALL
CP MONITOR SAMPLE ENABLE I/O CLASS DASD

CP MONITOR EVENT DISABLE ALL
CP MONITOR EVENT ENABLE PROC
CP MONITOR EVENT ENABLE STOR
CP MONITOR EVENT ENABLE USER ALL
CP MONITOR EVENT ENABLE I/O CLASS DASD
```

CP2KVMXT overhead can be additionally reduced by giving it the CP command authority to Q DASD ACTIVE so that it tracks only active dasd, rather than all defined dasd to HCPRIO.

## Saving Monitor Data in a File

If you want to save monitor information in a file, you will need a userid to connect to the monitor and use the MONWRITE utility to write out the data. This does not need to be the same userid that will, later, run the CP2KVMXT utility. It does not need anything higher than class G authority, but it will need special statements in its directory entry to be able to load the MONDCSS segment and connect to the monitor service. It needs a virtual storage size that will not conflict with MONDCSS, and it needs the following 2 statements to connect to the monitor:

```
IUCV *MONITOR MSGLIMIT 255
NAMESAVE MONDCSS
```

For more information on setting up a Monwrite userid, please refer to “Chapter 9. Monitoring Performance Using CP Monitor” in SC24-5999 z/VM Performance Guide (see Appendix B for the URL of the z/VM Publications library).

## Estimating the Space Needed for a Monitor Data File

The amount of space needed for the monitor data depends on many things, including the number of users, dasd devices, partitions, the complexity of the guests, and the length of the CP monitor interval. A table profiling 5 actual case studies is included below to help you, but the best way to estimate how much space the monitor data will be needed for your system is to collect data for 15 minutes and extrapolate from there.

| | Lx/WAS/Plex | Bank | University | IBM 7 VMs | VM/Linux |
|---|---|---|---|---|---|
| | 2084-303 | 9672-RA6 | 9672-RC6 | 9672-R96 | 2064-104 |
| Total RDEV | 785 | 616 | 128 | 2035 | 612 |
| Max Logged Users | 26 | 24 | 35 | 1035 | 25 |
| Real CPs | 12 | 1 | 3 | 9 | 8 |
| LCPs this image | 7 | 1 | 3 | 5 | 3 |
| partitions | 9 | 1 | 2 | 7 | 8 |
| Guests w/ multiple engines | 7 | 0 | 22 | 0 | 3 |
| # cache dev | 785 | 616 | 128 | 2035 | 612 |
| | | | | | |
| CP Monitor Interval (min) | 5 | 1 | 5 | 5 | 1 |
| collection period (hrs) | 0.5 | 0.2 | 23 | 1 | 0.1 |
| | | | | | |
| MB per interval | 0.7 | 0.4 | 0.2 | 1.7 | 0.6 |
| MB per hour | 8.8 | 22.0 | 2.4 | 20.2 | 35.3 |
| 4k blocks (actual) | 4742 | 1220 | 13927 | 5172 | 1206 |

## Setting Up CP2KVMXT:

To run the CP2KVMXT utility, you will need

- CP Class B (or equivalent) authority is required for the userid running CP2KVMXT, so that it has authority to issue the Q CHPIDS, Q CHPID *chpid* TYPE, and Q DASD ACTIVE commands.
- The number of lines of code executed has nearly doubled with the addition of BCU support. When running against realtime monitor data, it is even more important to minimize the monitor data collected and tune the running environment of CP2KVMXT. Please review the section entitled "Speeding up CP2KVMXT" in Appendix B if you will be running against realtime monitor data.
- A userid with authority to connect to the Monitor DCSS **OR** access to the saved monitor data file. To connect to the monitor, the userid needs statements in its directory entry that look something like this:

  IUCV *MONITOR MSGLIMIT 255
  NAMESAVE MONDCSS
- The disk space required to run the utility has increased considerably with the addition of BCU support. To estimate, divide the number of DASD rdevices in half to get the approximate number of 4k blocks of writable disk space required. For

example, a system with 612 DASD addresses would need 306 * 4k = 1.2meg of writeable disk space for a single run of the extract utility. This does *not* include the space for the monitor data itself.

- A reasonably large virtual storage size. 32meg should be enough, but more may be needed for a system with many DASD devices, and less than that may be fine too. When using realtime monitor data, the virtual storage size may have to be adjusted up or down so as not to conflict with the address of the MONDCSS.
- CP2KVMXT requires the Monview package, available from the VM Download packages website. See Appendix B for the URL.
- The CP2KVMXT code itself. See Appendix B for the URL of the repository.
- a VMPRF-like UCLASS file of user groups (see the section )
- Monitor data (see the sections on Monitor Requirements and Saving Monitor Data in a File)

## CP2KVMXT Installation

Please refer to installation instructions on the http://www.vm.ibm.com/download website and in the CP2KVMXT PACKAGE file for the most current information:

1) Download VMArc: http://www.vm.ibm.com/download/vmarc.module
2) Download Monview: http://www.vm.ibm.com/download/packages/monview.vmarc
3) Download CP2KVMXT: ftp://ftp.software.ibm.com/software/mktsupport/techdocs/cp2kvmxt.vmarc
4) In binary mode, upload vmarc.module, monview.vmarc, and cp2kvmxt.vmarc to the VM system where the monitor data resides.
5) On the VM system, PIPE < VMARC MODULE A | deblock cms | > VMARC MODULE A
6) VMARC UNPK MONVIEW VMARC fm = = outputFilemode (replace
7) VMARC UNPK CP2KVMXT VMARC fm = = outputFilemode (replace

## Grouping Users into Workloads (the UCLASS file):

In VM, "workloads" are groups of VM userids representing different business units, which will grow at different rates in the zCP3000 capacity planning model. By using the UCLASS file, you can group users into user-defined workload groups, like LINUXDB, MVS, NETWORK, LINUXWEB, VMSYS, etc

By default, the CP2KVMXT utility will look for a file with the same name as your output EDF file, but with the filetype of UCLASS. If it can't find this, it will use a file called GENERIC UCLASS to group user performance into workload groups. Users that don't fit any pattern found in the UCLASS file will form their own workload group, so you will potentially end up with many tiny workloads, more than your customer wants to see in a graph.

It is worth taking the time to tailor the groups so that they make sense to the people who will be seeing the results. When you tailor the UCLASS file, the extract utility has to be re-run in order to use the new workload grouping. Each time you run CP2KVMXT, a list of users active during that period, the UCLASS they were assigned to, and the CPU resource used by that individual userid, are logged in the ACTVUSRS file. You can use this to tailor your UCLASS grouping file into approximately even groups.

Alternatively, if your site runs VMPRF, you can start with the VMPRF UCLASS file and make adjustments. The VMPRF User Class Resource Utilization report (PRF014) will tell you if the workload groupings are reasonably granular. If you don't have a VMPRF UCLASS file, start with a list of logged on users, or a list of users provided by the system administrator from the directory mgmt software (DIRMAINT). Ensure that, in the UCLASS file,

- The userid is in columns 1-8 and does not contain embedded blanks or any special characters other than _ and $. A trailing * is allowed as a wildcard, i.e. LIN* in column 1 and LINUX in column 10 will accumulate work done by LINUPRD, LINTESTA, LINDBSVR, etc., userids into the LINUX workload.
- Workload group in columns 10-17; no embedded blanks or special characters except _ and $.
- Columns 18 and up are ignored by CP2KVMXT and can be used for comments.

Work with someone who is familiar with the work running on the system to group the userids into business unit based workloads. You may wish to do a short (1 hour) run of CP2KVMXT to see how these groupings turn out. CP2KVMXT will look for a UCLASS file with the same filename as the output EDF file, unless you specify otherwise on the CP2KVMXT command.

## Cookbook: Running CP2KVMXT with realtime monitor data:

There are 2 basic ways to run CP2KVMXT: using Monitor Data which has been written to a file, or reading data directly from the Monitor DCSS in main storage. The following procedure is for running CP2KVMXT realtime. The procedure for using saved monitor data follows this section.

**Note for first time users:** running against realtime monitor data can be the more complicated of the two methods, especially if the site does not already run the monitor. If tuning is needed, you will need to work closely with the site system programmer to size the dcss and try the various suggestions for optimizing CP2KVMXT's performance.

1. Review the setup requirements in the "Setting up CP2KVMXT" section.
2. Prepare the UCLASS file. Any userids found while processing the monitor data that are not covered by a userid or pattern in the UCLASS table, will go into the default group*OTHER*. Refer to the section **Grouping Users into Workloads** for detail.

3. Make sure that the userid which will run CP2KVMXT has been authorized to connect to the Monitor DCSS, and has Class B authority. The directory entry for that userid will need an authorizing statements like these:
   IUCV *MONITOR MSGLIMIT 255
   NAMESAVE MONDCSS
4. If there is a virtual storage addressing conflict with the Monitor DCSS, you will have to adjust the virtual storage size of the userid running CP2KVMXT. You will know this right away when you start running CP2KVMXT if you get the message DMSDCS343E Storage in range 01400000-02FFFFFF for MONDCSS in use. If this happens, try increasing the size of your virtual machine (using CP Q V STOR, then DEF STOR ___M). The max virtual storage setting in the directory entry may have to be increased to relieve the conflict.
5. If possible, have the userid running CP2KVMXT authorized to issue the CP command ***Q DASD ACTIVE***. CP2KVMXT will use less overhead if it monitors only active dasd and not all dasd addresses defined in HCPRIO. This is not a requirement, but if you get CP error message HCPMOV6274I, you may have to do this. Please refer to the Problem Determination section for more information.
6. If you will be using a parm file to supply runtime prompts, edit the CP2KVMXT PARMS file. The meaning of the various parms is explained in comments in the CP2KVMXT PARMS file. By default, CP2KVMXT will prompt for parameters. Parms supplied on the command line for input, output, and uclass fileid, will override the corresponding parm in the PARMS file.
7. If monitor data is not normally gathered, then you must enable and start the CP monitor from an E class userid. Please refer to the Monitor Requirements for detail. The monitor should run during core business hours, including both peak and non-peak times.
8. From the class E userid, you can CP Q MONITOR to see its status. You should see something like this:

```
MONITOR EVENT ACTIVE    BLOCK    4     PARTITION    3584
MONITOR DCSS NAME - MONDCSS
CONFIGURATION SIZE       68 LIMIT        1 MINUTES
CONFIGURATION AREA IS FREE
USERS CONNECTED TO *MONITOR - MONWRITE
MONITOR   DOMAIN ENABLED
PROCESSOR DOMAIN ENABLED
STORAGE   DOMAIN ENABLED
SCHEDULER DOMAIN DISABLED
SEEKS     DOMAIN DISABLED
USER      DOMAIN ENABLED
  ALL USERS ENABLED
 I/O       DOMAIN ENABLED
THE FOLLOWING DEVICES ARE ENABLED:
2000-203F  2300-247F  2540-257F  2600-263F  2C00-2EFF  4000-427F
4300-44FF  4800-4848  4880-48C8  4900-4948  4980-49C8  4A00-4BFF
5000-50FF  5300-53FF  6300-637F  6400-647F  7000-77FF  8420-842F
8440-844F  AC00-AFFF  B300-B3FF  B700-B7FF  B900-BAFF  BD00-BDFF      BF00-C1FF
D800-D853  D900-D953  E900-E93F
```

```
APPLDATA  DOMAIN DISABLED
MONITOR SAMPLE ACTIVE
      INTERVAL    5 MINUTES
      RATE     1.00 SECONDS
MONITOR DCSS NAME - MONDCSS
CONFIGURATION SIZE      241 LIMIT        1 MINUTES
CONFIGURATION AREA IS FREE
USERS CONNECTED TO *MONITOR - MONWRITE
MONITOR   DOMAIN ENABLED
SYSTEM    DOMAIN ENABLED
PROCESSOR DOMAIN ENABLED
STORAGE   DOMAIN ENABLED
USER      DOMAIN ENABLED
     ALL USERS ENABLED
I/O       DOMAIN ENABLED
THE FOLLOWING DEVICES ARE ENABLED:
2000-203F  2300-247F  2540-257F  2600-263F  2C00-2EFF  4000-427F
4300-44FF  4800-4848  4880-48C8  4900-4948  4980-49C8  4A00-4BFF
5000-50FF  5300-53FF  6300-637F  6400-647F  7000-77FF  8420-842F
8440-844F  AC00-AFFF  B300-B3FF  B700-B7FF  B900-BAFF  BD00-BDFF
BF00-C1FF  D800-D853  D900-D953  E900-E93F
APPLDATA  DOMAIN DISABLED
Ready; T=0.01/0.04 20:26:08
```

9. Start the CP2KVMXT utility to create an EDF file:
   a. Refer to Command Syntax later in this doc for syntax. In the parmfile or as an argument to CP2KVMXT, you should specify MONDCSS in place of the input monitor data fileid. Even if your MONDCSS is called something else, specify MONDCSS as the input.
   b. If you specified that CP2KVMXT use the parameter file by using the PARMFILE option, it will pick up runtime parameters from there and override them with any command line options. Otherwise, you will see several prompts, the first of which is to enter a title for the study, including the customer name and perhaps a date.
   c. If you did not specify the CP2KVMXT parm file, the utility will ask you for a **DURATION**. This is the period of time that a SAMPS record in ZCP3000 will represent. You should pick a value which is a multiple of the CP Monitor interval and will generate a reasonable number of data points on a utilization graph (for example, 4 hours of data and a duration of 15 minutes would generate a graph with 16 data points). For an 8 hour measurement period, 30 minutes would be a good value for the **DURATION**.
   d. If you did not specify the CP2KVMXT parm file, and if there is more than one model for that n-way machine, you may be prompted at the end of the first duration to provide the correct model number of the machine being measured. Once you see an "end of duration" message go by, there should be no further prompts. .

10. Progress messages are issued at interval and duration ends, and after every 5000 records processed. The utility will end with a message about where the new EDF file was written. If you want to end CP2KVMXT earlier than the time you specified (via prompt or parmfile), you can do this by hitting the <enter> key once, letting it sit in VM READ for 2 minutes, and then hitting the <enter key> again. This will time out the MONDCSS connection. Entering HX will also work, but will likely abend CMS and require that the userid re-ipl.

11. If the final EDF was not created for some reason (i.e., you HX'd it), you can create an EDF from the intermediate results file with the following command.
    **CP2KVMXT SampsFileid ( restart**
    Refer to the CP2KVMXT Command Syntax section for more detail.

13. Send the EDF file to the IBMer or Business Partner performing the Capacity Analysis.

## Cookbook: Running CP2KVMXT with saved monitor data:

There are 2 basic ways to run CP2KVMXT: using Monitor Data which has been written to a file, or reading data directly from the Monitor DCSS in main storage. The following procedure is for using saved monitor data. The procedure for running CP2KVMXT with realtime data precedes this section.

The zCP3000 modeling tool includes a utility to merge EDFs for a single system from different days. This enables you to collect data for only the business hours. Because ZCP3000 uses 90th percentile rather than average for planning, however, it is still recommended to collect data for a longer period of time than just the expected business hours.
**Note for first time users:** running against saved monitor data is often the simpler of the two methods, as long as sufficient disk space for the monitor file is available.

1. Review the setup requirements in the "Setting up CP2KVMXT" section.
2. Prepare the UCLASS file. Any userids found while processing the monitor data that are not covered by a userid or pattern in the UCLASS table, will go into the default group*OTHER*. Refer to the section **Grouping Users into Workloads** for detail.
3. If monitor data is not normally gathered, then you must prepare the MONWRITE machine. Refer to the section Saving Monitor Data in a File for detail.
4. If monitor data is not normally gathered, then you must enable and start the CP monitor from an E class userid. Please refer to the Monitor Requirements for detail. The monitor should run during core business hours, including both peak and non-peak times. Typically, the monitor is started and stopped from a scheduler program such as WAKEUP, running on an operations utility userid.
5. Once the monitor is enabled, start up the MONWRITE machine, so that it can write monitor data out from the monitor DCSS into a file. Typically, the monwrite machine

is started and stopped by the same WAKEUP machine used to start and stop the monitor.

6. Using the same process as when running CP2KVMXT with realtime monitor data, CP Q MONITOR from any Class B enabled userid to see its status.
7. If you are not using a WAKEUP machine to automate the process, then you must manually stop the monitor with this command so that it can gracefully finish writing out interval data: CP MONITOR STOP
   **Note:** The monitor needs to be stopped only for an instant to complete the Monitor Data file. You can start the monitor back up again right away for other applications that use it (i.e., VMRTM or FCONX).
8. From the userid which will run CP2KVMXT, link to the Monwrite disk. Make sure the Monview and CP2KVMXT program files and the customer's UCLASS file are present (they can be on any accessed disk).Start the CP2KVMXT utility. Refer to Command Syntax later in this doc for syntax. In the parmfile or as the first argument to CP2KVMXT, you should specify the input monitor data fileid. This too can be automated by a WAKEUP type utility userid, to start up CP2KVMXT once the monitor has been stopped.
9. Progress messages are issued at interval and duration ends, and after every 5000 records processed. The utility will end with a message about where the new EDF file was written.
10. Using the same process as when running CP2KVMXT with realtime monitor data, you can create an EDF from the intermediate results if the final EDF was not created for some reason.
11. Send the EDF file to the IBMer or Business Partner performing the Capacity Analysis

# .Appendix A: Command and Parmfile Syntax:

By default, the output file will have the same *filename* as the input monitor data, with a filetype of EDF. The output EDF file will be normally be written on the first r/w mdisk it finds. The CP2KVMXT PARMS file allows the specification of variables and parameters that would otherwise have to be supplied either as an argument or in response to a prompt. Comments in the sample CP2KVMXT PARMS file show the syntax, which basically is the keyword followed by a blank-delimited = sign, then the value for that keyword, with non-numerics in double quotes.

To process a monitor data file

**CP2KVMXT mondataFileid outputFileid ( uclassFileid  options**

To process realtime monitor data:

**CP2KVMXT MONDCSS outputFileid ( uclassFileid  options**

If the monitor data, uclass file, and outputFileid have the same filename, when monitor records before 10am should be ignored, and only 8 durations should be processed:

**CP2KVMXT mondataFn  ( STARTTIME 10:00 DURLIMIT 8**

If all runtime parms have been specified in CP2KVMXT PARMS:

**CP2KVMXT ( PARMFILE**

To use the CP2KVMXT PARMS but override the fileids specified in it:

**CP2KVMXT  mondataFileid outputFileid ( uclassFileid PARMFILE**

To process build an EDF from an intermediate results file:

**CP2KVMXT SampsFileid outputFileid ( restart**

| Parm/Option | Description: | Where used: |
|---|---|---|
| **MondataFileid** | is the fn ft fm of the monitor data file. This should be a fixed blocked file with a block size of 4096. | Command line parameter PARMFILE keyword |
| **OutputFileid** | is where the EDF should be written. The output fn and fm default to be the same as that of the MondataFileid. The default ft is “EDF”. | Command line parameter PARMFILE keyword |
| **UclassFileid** | is the file containing the grouping of users into workloads. The uclass fn and fm default to be the same as that of the MondataFileid. The default ft is “UCLASS”. | Command line option PARMFILE keyword |
| **SampsFileid** | is the fn ft fm of the intermediate checkpoint file. It always has a filetype of SAMPS, and uses the filename of the specified output edf fileid. | Command line parameter o |
| **PARMFILE** | means that CP2KVMXT will look in the CP2KVMXT PARMS file for run parameters. | Command line option only |

| | | |
|---|---|---|
| **restart** | Is a keyword indicating that the EDF should be built from an existing intermediate results file. | Command line parameter o |
| **Startdate** mm/dd/yy | Skip all activity sample records in the monitor data until this date. Default is to use the first date found in the monitor data. *Incompatible with MONDCSS realtime input* | Command line option PARMFILE keyword |
| **Starttime** hh:mm | Skip all activity sample records in the monitor data until this time. Default is to use the first samples found in the monitor data. *Incompatible with MONDCSS realtime input* | Command line option PARMFILE keyword |
| **Durlimit n** | Process only this number of durations of data. | Command line option PARMFILE keyword |
| **Runhours n** | Connect to and process realtime monitor data for this length of time. | PARMFILE keyword only |
| **Enterprise** | Title identifying this customer and set of monitor data. | PARMFILE keyword only |
| **CPType** | IFL or CP; the type of engine this VM image runs on. This is ignored for z/VM 4.4 and above running on zSeries because the information is in the monitor data. | PARMFILE keyword |

## Appendix B: Problem Determination

1. **Symptom**: vmarc won't work. **Explanation:** possibly something went wrong in the file transfer. Upload it again, and make sure it is being transferred in binary mode**. It should end up with record length=80 (after the PIPE deblock).**
2. **Symptom**: vmarc says something is wrong with monview.vmarc or cp2kvmxt.vmarc. **Explanation:** possibly something went wrong in the file transfer. Upload them again, and make sure it is being transferred in binary mode. **VMARC archived files should have a record length=80.**
3. **Symptom**: DMSDCS343E Storage in range 01400000-02FFFFFF for MONDCSS in use. **Explanation:** You must redefine the size of your virtual storage so that it does not conflict with the MONDCSS segment. I.e., in the above example, the user's virtual storage size was set to 32M, and MONDCSS tried to load into the area from 20-50M. DEF STOR 64M then re-ipling CMS fixed the problem.
4. **Symptom**: FPLSMG319E Not authorized to communicate with *MONITOR or DMSDCS283E The MONDCSS saved segment could not be loaded; return code 449 from SEGMENT LOAD
   **Explanation:** Your userid's directory entry needs 2 statements like this:
   IUCV *MONITOR MSGLIMIT 255
   NAMESAVE MONDCSS[1]
5. **Symptom**: HCPMOV6274I The sample data messages and corresponding records have been purged. FPLSMG313E IPRCODE Message was purged received on IUCV instruction **Explanation:** CP2KVMXT could not keep up with the monitor. Refer to "Speeding up CP2KVMXT" following this section for suggestions of things you can do to speed things up.
6. **Symptom**: **"Insufficient storage" or "Machine storage exhausted";** this means that there is not enough virtual storage defined for this userid to run the extract. Use the DEF STOR xxM command to define a larger virtual storage size. If your userid is already at the max, you will need to have the system administrator authorize you for a larger virtual storage size.
7. **Symptom**: cp2kvmxt completes, but no EDF file is created. **Explanation:** cp2kvmxt probably did not complete successfully. If a file with a filetype of **DEBUG** exists on the A disk, please send it to Gretchen Frye at the address listed in Appendix B, along with an explanation of the problem. Checkpoint data is kept as the utility runs, so it is possible to generate an EDF from the part of the monitor data that was successfully processed. Use the restart option to build an EDF from intermediate results (see above).

[1]The NAMESAVE statement is needed if the MONDCSS was defined as a restricted segment.

8. **Symptom**: cp2kvmxt completes, but no EDF file is created, and it seems to take a long time doing something after the last duration has been processed. **Explanation:** It may have failed in the step that creates the EDF out of the checkpoint data. If the SAMPS file is very large, it may run out of virtual storage when processing the checkpoint data. If the SAMPS and BCUDATA files were created (check the timestamp), and you can see that the checkpoint is large (ie, larger than the DEBUG file), try the restart option: CP2KVMXT *fn* SAMPS *fm* ( RESTART If that doesn't work, try increasing the virtual storage of the machine and use the restart option again. If that doesn't work, then it may be necessary to re-run CP2KVMXT and use the start and end parameters to create multiple, smaller EDFs.
9. **Symptom**: cp2kvmxt completes, but no EDF file is created, and no DEBUG file was created. **Explanation:** the buildrec rexx file may be flawed. A bad copy of CP2KVMXT.VMARC was on the VM download site for a while. To see if this is the problem, browse or edit the BUILDREC REXX file; if it looks like compiled code, then that is the problem. Download a new copy of the cp2kvmxt.vmarc and reinstall.
10. **Symptom**: cp2kvmxt gets a rexx error and does not complete. **Explanation:** oh, lots of reasons, mostly due to seeing something unexpected in the data. Make sure the monitor file you are running against is actually a CP Monitor file. Monitor data files are in hex, in a FB4096 file. The records wrap, so there's not really anything you can look for, but it is supposed to look unreadable. If there is a rexx error that makes the utility end abnormally, a debug file with a filetype of DEBUG will be generated. Please attach that to an email and send to the author, Gretchen Frye, whose email address is listed in Appendix B.
11. **Symptom**: i/o counts don't look right. **Explanation:** If the numbers are plausible (ie, not a negative number, or an impossibly large value), then it is possible that the difference is in understanding what the field actually means. Some numbers are based on logical requests, for example, WIO and WEXCPV. Other values count Start SubChannel commands, for example, the SYS DASDIO attribute. In VMPRF, some IO counts are for all devices (the ones on the PRF002 report), while others are for dasd class only. Several numbers, like WIOV (vector of SSCH by workload), are not provided by the CP Monitor. Instead, these numbers are estimated from related data; for example, SSCH by workload is derived from the workload's proportion of logical I/Os and the system total SSCH + RSCH.
12. **Symptom**: duration lengths in the **SAMP** records are not all the same, and are not what I specified for a duration; **Explanation:** the duration has to be ended when the first CP monitor interval record (domain 1 record B) past duration end occurs, because that's when CP collects the sample data.
13. **Symptom**: why don't the summed WIO numbers equal the system DASDIO number? **Explanation:** the WIO number is based on DASD I/O requests (logical IO), while the system dasdio number is based on total SSCH + RSCH (physical IO). The WIO number is actually the average value found in WEXCPV. The values in the WIOV vector are derived from the system dasdio SSCH+RSCH and, when summed, should be within rounding error of the system dasdio number. I have seen a problem

with timestamps being out of order that exaggerates this problem, however, and I don't have a good solution for it so far. If you think this is happening, contact the author for further problem analysis.

## Speeding up CP2KVMXT:

This section applies both to realtime MONDCSS input and saved monitor data files, but it is much more critical for realtime input because the connection to the monitor ends if CP2KVMXT can't keep up with the monitor. For MONDCSS input, after reading each set of records, CP2KVMXT replies to the monitor service that it has read them. If it does not reply by the time the monitor service wants to reuse those pages, the monitor takes the pages and issues an HCPMOV6274I error message. The CMS Pipeline connection to the monitor ends the connection when this happens, issuing the FPLSMG313E error message. For file input, the CP2KVMXT processing is more like a batch job, in that it will just take longer to run. If you need the data sooner rather than later, however, the same things will speed up CP2KVMXT processing of both realtime and saved file data.

1. Limit CP2KVMXT to active dasd only by arranging to have your userid given sufficient authority to issue the Q DASD ACTIVE command. This requires class B authority command, which some sites may be hesitant to authorize you for. Alternatively, they can reclassify the CP query command (i.e., as class Y) and give class Y authority to your userid. Refer to "Redefining CP Command Privilege Classes" in the CP Planning and Administration guide, at the z/VM Internet Library http://www.vm.ibm.com/library/

2. Restrict what the monitor is enabled for. Enable for only DASD class I/O, or for any specified device address ranges. Even if not online, tables are built for all devices that monitoring is enabled for (unless Q DASD ACTIVE was used), so it can make a difference. **Make sure SCHEDULER and SEEKS are disabled** unless there is a specific reason for enabling them, and even then, you should enable for a subset of users or rdevs, instead of all.

3. Ensure that the MONDCSS is the right size for your system. The default size of 3MB may not be big enough, especially for guest-hosting systems with lots of DASD. Follow the guidelines for sizing MONDCSS in the z/VM Performance Guide on http://www.vm.ibm.com/library/

4. If your site has the Rexx compiler (REXXD EXEC is the compiler), try renaming and compiling BUILDREC REXX. The compiled BUILDREC must have a filetype of REXX in order to execute. It won't make any difference to compile CP2KVMXT, because it is mainly a front end and does not execute much code.

5. Have the CP2KVMXT userid bumped up in the dispatch list, by increasing relative share. Using **OPTION QUICKDSP** in the directory entry or the SET QUICKDSP ON command will only help if the system actually carries an eligible list in addition to the dispatch list.

6. From the HCPMOV6274I data, note whether the problem occurred with "event" or "sample" data. Increase the size and limit of the config on the CP Monitor command for "event" or "sample" so that the records are retained a little longer.

7. Consider increasing the CP Monitor interval size. This will reduce the sampling frequency, and thereby reduce the amount of data collected.

# Appendix C: EDFI Field Descriptions

| EDFI - Enterprise Data File for Input | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| **Head** | **Header Section** | **Type (length)** | **Domain / Rec** | **VM Source Fieldname** | **MVS equivalent** |
| ENT= | Enterprise Name | char(50) | - | User | User |
| Source= | Version which produced this file | Char | - | generated | |
| | | | | | |

| **CEC** | **CEC Section** | **Type (length)** | **Domain / Rec** | **VM Source Fieldname** | **MVS equivalent** |
|---|---|---|---|---|---|
| CECID= | CEC Identifier | Char | - | User | User |
| SUPVR= | Supervisor | Char | - | generated | |
| CPUMOD= | CPU Model | Char | D1R4 | MTRSYS_SYSMTYPE, user prompt | SMF70MOD SMF70VER |
| VC= | Version Code | Char | D1R5 | MTRPRP_PFXIDVER | SMF70VER |
| PR= | CP + IFL + ICF | Number | D0R17 | SYTCUM_LCUPCPCT | SMF70BNP |
| SR= | CPU serial # | Number | D1R5 | MTRPRP_PFXIDSER | SMF70SER |
| PRV= | Only if PR changes | Vector | D0R17 | SYTCUM_LCUPCPCT | SMF70BNP |
| CPV= | Traditional engines | Vector | D0R19 | SYTSYG_CPUCFGCT | |
| IFLV= | IFL engines | Vector | derived | PRV - CPV | |
| ICFV= | Coupling Facility | Vector | n/a | Merge in from MVS EDF | |
| CPUMODV= | CPU Model Names | Multiple text strings | D1R4 | MTRSYS_SYSMTYPE, user prompt | SMF70MOD SMF70VER |
| CMIND= | Index of current Vector CPC in CPUMODV | Vector | | | |

| SYS | System Image | Type (length) | Domain / Rec | VM Source Fieldname | MVS equivalent |
|---|---|---|---|---|---|
| SYSID = | System ID | Char | D1R4 | MTRSYS_SYSTMID | SMF70SID |
| SCP= | SCP | Char | D1R4 | MTRSYS_HCPCPEPP | SMF70MVS |
| VERSION= | SCP level | Char | D1R4 | MTRSYS_HCPCPEID | SMF70RLS |
| NSAMPS= | Number of samples | Number | - | generated | |
| WC= | Wait Complete | 1 or 0 | D0R16 | SYTCUP_LCUCWCPL | |
| BIT= | 64 bit mode indicator | 1 (appears only if in 64 bit mode0 | D1R4 | MTRSYS_CALESAME | SMF70EME |
| RMFINTL= | CP monitor interval | Number (min) | D1R9 | MTRSPR_INTERVAL | |
| GMTOF= | offset GMT | +/- hh:mm | D1R4 | MTRSYS_SYSZONE | |
| DASDIO= | Dasd SSCH rate | Number | D1R6 D6R3 | MTRDEV_RDEVCLAS, IODDEV_SCMSSCH | SMF74SSC |
| PAGE= | Page Operations | Number | D0R1 | SYTSYP_PLSIOPR + PLSIOPW | SMF75SIO |
| CS= | CS installed MB | Number | D1R7 | MTRMEM_RSAGSTOR | SMF71TFC + SMF71FIN |
| CSAVAIL= | | | | MTRMEM_SYSGTORS | SMF71TFC + SMF71FIN |
| ES= | ES installed MB | Number | D1R7 | | SMF71OLE |
| ESAVAIL= | | | | MTRMEM_SYSGTORS | SMF71TFC + SMF71FIN |
| SCPCS= | Total Resident Nucleus | | D1R7 | MTRMEM_HCPMM4 | SMF71FIN+ASR+ALP+AVP |
| PAGEV= | Page Operations | Vector | D0R1 | SYTSYP_PLSIOPR + PLSIOPW | SMF75SIO |
| | | | | | |
| PGTOES= | Total pages to ES | Vector | D0R5 | SYTXSP_PFXPGIN + PLSPGIN | SMF71PES |
| PGFROMES= | Pages from ES | Vector | D0R5 | SYTXSP_PLSPGOUT | SMF71RES |
| DASDIOV= | Dasd SSCH rate | Vector | D1R6 D6R3 | MTRDEV_RDEVCLAS, IODDEV_SCMSSCH | SMF74SSC |
| LPARNO= | index of this sysid in the LPAR array | Number | D0R16 | SYTCUP_CALPTIS | SMF70PTN |
| LPAR= | Partition names | Vector | D0R16 | SYTCUP_LCUPNAME | SMF70LPM |
| LPPRCn= | avg # CPs this partition | Vector | D0R16 | SYTCUP_LCUPCPCT | SMF70ONT |
| LPPCTMn= | seconds of CPU time this partition | Vector | D0R16 | SYTCUP_LCUCACTM | SMF70PDT |
| LPPRIn= | avg # ICFs this partition | Vector | D0R16 | SYTCUP_LCUPCPCT | SMF70ONT |
| LPPITMn= | seconds of ICF time this partition | Vector | D0R16 | SYTCUP_LCUCACTM | SMF70PDT |
| LPPRLn= | avg # IFLs this partition | Vector | D0R16 | SYTCUP_LCUPCPCT | SMF70ONT |
| LPPLTMn= | seconds of IFL time this partition | Vector | D0R16 | SYTCUP_LCUCACTM | SMF70PDT |
| LPWGTn= | avg LPAR wgts this partition | Vector | D0R16 | SYTCUP_LCUCWGHT | SMF70MIS, SMF70BPS, SMF70MAS |

| | | | | | |
|---|---|---|---|---|---|
| LPCAPn= | is partition capped? (1 or 0) | Vector | D0R16 | SYTCUP_LCUCCAPP | SMF70VPF, SMF70CAP |
| CECUTILV= | Utilization for the image | Vector | D0R16 | SYTCUP_LCUCACTM | SMFWAT, SMF70PDT |

| SAMP | System Samples | Type (length) | Domain / Rec | VM Source Fieldname | MVS equivalent |
|---|---|---|---|---|---|
| DUR= | Duration (hh:mm) | Char | - | User | User |
| SIO= | Sample I/O Rate | num | D0R1 | SYTSYP_PLSCTSS | |
| SPAGE= | Sample Page Rate | num | D0R1 | SYTSYP_PLSPIOPR+IOPW | |
| | | | | | |
| | | | | | |

| WORKS | Workload Samples | Type (length) | Domain / Rec | VM Source Fieldname | MVS equivalent |
|---|---|---|---|---|---|
| WDESC= | Description | char | - | from UCLASS file | user parm |
| WIO= | Avg dasd IO requests | num | - | derived from system SSCH and user VDSCT | SMF72ITS |
| WCS= | Central Storage | num MB | D4R3 | USEACT_VMDCTPVR | SMF72FT1,2 |
| WES= | Expanded Storage | num MB | D4R3 | USEACT_VMDCTPVR | SMF72ER1,2 |
| WPAGE= | Demand Page Rage | page/sec | D4R3 | USEACT_CALCPPGR+CALCPPGW | RMF72PIN |
| | | | | | |
| WTRANSRS= | Trivial+nontrivial | rate/second | D4R8 | avg of TRANV | RMF72TTX |
| WRESP= | Response Time | seconds | D4R8 | avg of RESPV | SMF72TTM |
| TRANV= | Transaction rate | vector | D4R8 | count of d4r8 / elapsed | SMF72TTX |
| RESPV= | Response Time | vector | D4R8 | USETRE_VMDSUSCK-VMDMTTOD | SMF72TTM |
| WIOV= | SSCH count | vector | calc | IODDEV_SCMSSCH | SMF72IRC+R723CIRC |
| WPAGEV | Wkld Paging | vector | D4R3 | USEACT_CALCPPGR+CALCPPGW | SMF72PIN+R723CPIN |
| WCPUTM | Wkld CPU seconds | vector | D4R3 | USEACT_VMDTTIME | SMF72CTS+SMF72STS+R723Cnnn |
| WCSV | Wkld Central Stor | vector | D4R3 | USEACT_VMDCTPVR | SMF72FT1,2 |
| WESV | Wkld Expnd Stor | vector | D4R3 | USEACT_VMDCTPVR | SMF72ER1,2 |
| WEXCPV | Wkld Dasd IO Requests | vector | D4R3 | USEACT_VMDVDSCT | SMF72ITS+R723CIOC |

| BCU | Basic Configurable Unit for DASD | Type (length) | Domain / Rec | VM Source Fieldname | MVS equivalent |
|---|---|---|---|---|---|
| BCUID | BCU Identification | char | D6R4 | IODCAD_CALDATA | User Parm |
| CTYPE= | CU type | char | D1R6 | MTRDEV_RDEVCUID, MTRDEV_RDEVCUMN | User Parm/ SMF74CU |
| BCUDASD1= | Dasd type (online devices only) | char | D1R6 | MTRDEV_RDEVDVID | User Parm/ SMF74DEV |
| BCUDASDN= | # units this type (online devices only) | num | D6R3 | counted | computed |
| CACHE= | Available cache | num (MB) | D6R4 | device hardware | UserParm/ CSCONF |
| NVS= | Non-volatile storage | num (MB) | D6R4 | device hardware | UserParm/ CSCONF |
| NOAD= | # addresses under this BCU | num | D6R3 | counted | From BCU MAP |
| BCUIO= | Total I/O Rate | num[2] | D6R3 | IODDEV_SCMSSCH | SMF74SSC |
| BCURESP= | Average Response Time | num | | computed | computed |
| BCUCONN= | Average Connect Time | num | D6R3 | IODDEV_SCMCNTIM | SMF74CNN |
| BCUDISC= | Average Disconnect Time | num | D6R3 | IODDEV_SCMDDTIM | SMF74DIS |
| BCUPEND= | Average Pend Time | num | D6R3 | IODDEV_SCMFPTIM | SMF74PEN |
| BCUQUE= | Average IOS Queue Time | num | D6R3 | IODDEV_HFCTIO | SMF74QUE |
| BCUSKEW= | Maximum device busy to average . | num | | computed | computed |
| BCUR= | Read/Write Ratio[3] | num | D6R4 | computed | CRR:"Total (Cache) R/W Ratio" |
| BCUH= | Ratio: Read Hits[4] / All Reads | 0< num <1 | D6R4 | IODCAD_CALDATA | CRR:"Total Read H/R" |
| BCUW= | Ratio: Fast Write[5] Hits / All Writes | 0< num <1 | D6R4 | IODCAD_CALDATA | CRR:"Total F/W H/R" |
| BCUG= | Ratio: Sequential DASD to Cache / All I/O | 0< num <1 | D6R4 | IODCAD_CALDATA | CRR: "DASD to Cache Transfers - Sequential" divided by "Total I/O Requests" |
| BCUIOV= | I/O Rate per sample. | vector | D6R3 | IODDEV_SCMSSCH | SMF74SSC |
| BCURESPV= | Avg Response | vector | D6R3 | computed | computed |

[2]Time fields in the BCU are in seconds.
[3]BCU time fields include all online devices under that BCU, whether they are cached or not.
[4]Cache Reads include Search/Read requests for Normal, Sequential, and Fast Write I/Os.
[5]Cache Writes include Write requests for Normal, Sequential, and Fast Write I/Os

| | | | | | |
|---|---|---|---|---|---|
| | Time per sample | | | | |
| BCUCONNV= | Avg Connect Time per sample | vector | D6R3 | IODDEV_SCMCNTIM | SMF74CNN |
| BCUDISCV= | Avg Disconnect Time per sample | vector | D6R3 | IODDEV_SCMDDTIM | SMF74DIS |
| BCUPENDV= | Avg Pend Time per sample | vector | D6R3 | IODDEV_SCMFPTIM | SMF74PEN |
| BCUIOSQV= | Avg IOS Queue Time per sample | vector | D6R3 | IODDEV_HFCTIO | SMF74QUE |
| BCURV= | Read Write Ratio per sample | vector | D6R4 | computed | CRR:"Total (Cache) R/W Ratio" |
| BCUHV= | Read Hit Ratio per sample | vector | D6R4 | IODCAD_CALDATA | CRR:"Total Read H/R" |
| BCUWV= | Fast Write Hit Ratio per sample | vector | D6R4 | IODCAD_CALDATA | CRR:"Total F/W H/R" |
| BCUGV= | Sequential Stage Ratio per sample. | vector | D6R4 | IODCAD_CALDATA | CRR: "DASD to Cache Transfers - Sequential" divided by "Total I/O Requests" |

| **PATH** | **BCU Path Data** | **Type (length)** | **Domain / Rec** | **VM Source Fieldname** | **MVS equivalent** |
|---|---|---|---|---|---|
| PID= | Path ID | Char chpid | D6R3 | IODDEV_RDEVLPM | User |
| PTYPE= | Channel Type | Char 'P' parallel, 'E' ESCON, 'F' FICON | | class B CP command : Q CHPID *chpid* TYPE | SMF73ACR |
| PBUSYV= | Path Busy | vector | D0R9 | SYTCPC_HFCHBUSY | SMF73BSY |

| **ACT** | Actuator Data | **Type (length)** | **Domain / Rec** | **VM Source Fieldname** | **MVS equivalent** |
|---|---|---|---|---|---|
| SID= | Subchannel ID | num (hex) | D6R3 | IODDEV_RDEVLPM | User |
| V= | Volser[6] | Char | D6R3 | IODDEV_RDEVSER | SMF74SER |
| A= | Address | num (hex) | D1R6 | MTRDEV_RDEVDEV | SMF74NUM |
| T= | DASD Type | char | D1R6 | MTRDEV_RDEVCUID, MTRDEV_RDEVCUMN | SMF74DEV |
| R= | I/O Rate | num | D6R3 | IODDEV_SCMSSCH | SMF74SSC |
| SDS= | standard deviation for service | num | D6R3 | computed | computed |
| Q= | IOSQ | num | D6R3 | IODDEV_HFCTIO | computed |
| P= | Pend Time | num | D6R3 | IODDEV_RDEVFPTIM | SMF74PEN |
| D= | Disconnect Time | num | D6R3 | IODDEV_RDEVDDTIM | SMF74DIS |
| C= | Connect Time | num | D6R3 | IODDEV_RDEVCNTIM | SMF74CNN |

[6]Volser is not unique, since a device attached to a guest may have any volser (or null).

| | | | | | |
|---|---|---|---|---|---|
| DS= | Minidisks Defined | num | D6R3 | IODDEV_RDEVLCNT | SMF74NDA |
| SDR= | standard deviation for response | num | D6R3 | computed | computed |
| RWR= | Read Write Ratio | num | D6R3 | computed | CRR:"Total (Cache) R/W Ratio" |
| RDHT= | Read Hits / All Reads | num | D6R4 | IODCAD_CALDATA | CRR:"Total Read H/R" |
| FWHT= | Fast Write Hits / All Writes | num | D6R4 | IODCAD_CALDATA | CRR:"Total F/W H/R" |
| SQSTG= | Sequential DASD to Cache / All I/O | num | D6R4 | IODCAD_CALDATA | CRR: "DASD to Cache Transfers - Sequential" divided by "Total I/O Requests" |
| PC= | Percent Cacheable | num | D6R3 | IODCAD_CALDATA | CRR: "Total Cacheable I/Os" divided by "Total I/O Requests" |

# Appendix D: Resources

## Websites:

IBM VM Download Packages
http://www.vm.ibm.com/download/packages/

How to download a VMARC type package from VM Downloads
http://www.vm.ibm.com/download/#downvmarc

IBM WSC - Capacity Planning Support Tools menu (IBM Internal):
http://w3.ibm.com/support/americas/wsc/cpsproducts.html

IBM WSC - Capacity Planning Support Tools menu (IBM Business Partners):
http://partners.boulder.ibm.com/src/atsmastr.nsf/WebIndex/PRS1762

## Publications:

The CP2KVMXT user's guide (that you are reading right now) can be found at:
http://www-1.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS720

z/VM Internet Library
http://www.vm.ibm.com/library/

VMPRF V1R2.2 User's Guide and Reference
http://www.vm.ibm.com/pubs/pdf/fpra1a04.pdf

z/VM V4R3: CP Command and Utility Reference
http://www.vm.ibm.com/pubs/pdf/hcsg0a00.pdf

## Contacts:

For questions and general support: Bernice Riley
Internet email address: cpstools@us.ibm.com
Normal US business hours (EST): 301-240-2645

For technical issues: Gretchen Frye
Internet email address: frye@us.ibm.com
IBM VM address: frye at wscvm

**Note:** Gretchen's VM address is for debug files and data ONLY. If sending data through email, please VMARC or PKZIP it first.

Return to the top