CA Performance Management for OpenVMS

SDK Guide

r3.1

Contact Technical Support

For online technical assistance and a complete list of locations, primary service hours, and telephone numbers, contact Technical Support at http://ca.com/support.

Contents

Contents 5

Chapter 1: Performance Manager Callable Interface 7

How You Use This Guide 7
Related Documentation 8

Chapter 2: Performance Manager APIs 9

Data Type Identifiers 9
Sample Code 10
PSPA$OPEN_CONTEXT 12
PSPA$READ_CONTEXT 37
PSPA$CLOSE_CONTEXT 53

Chapter 3: PSPA Libraries 57

Compile and Link with the PSPA MACRO Library 57
PSPA Record Field Definitions 58
- Configuration Record 59
- Process Record 60
- Metrics Record 64
- CPU Record 71
- Parameter Record 71
- Time Record 75
- Device Record 75
- Tape Record 78
- Communications Record 79
- Hot File 79

Appendix A: Performance Manager Sample Programs 81

Index 83

Chapter 1: Performance Manager Callable Interface

This guide provides detailed information about the CA Performance Management for OpenVMS Software Development Kit (SDK). It is intended for application support personnel and application developers who maintain and administer their company's OpenVMS systems. Use the SDK to build application programming interfaces (APIs) for managing the Performance Manager.

We suggest that you read the *CA Performance Management for OpenVMS Performance Manager Administrator Guide* before you continue with this guide. If you are responsible for the installation of CA Performance Management for OpenVMS, you should also read the *CA Performance Management for OpenVMS Installation Guide* before continuing.

This section contains the following topics:

How You Use This Guide (see page 7)
Related Documentation (see page 8)

How You Use This Guide

The *SDK Guide* assumes that you are familiar with the OpenVMS operating system and some system management functions such as the use of system privileges. This guide also assumes that you are familiar with processing Digital Command Language (DCL) commands, in both interactive and batch modes, and HP C language rules.

An Application Program Interface (API) is a tool that enables you to access daily Performance Manager data. Within the documentation set for this product, you might see the following terms that are synonymous with API:

- Callable Interface for Data Extraction (CIFDE)
- Callable interface
- SDK callable interface

With the APIs, you can access data in real-time mode by either accessing the data currently being written to a daily data file or by initiating a dedicated real-time data collector on a remote node.

The arguments in this guide follow the conventions in HP's *OpenVMS Calling Standard* guide. Data extraction procedures are called and Performance Manager record fields are referenced with a user-written program that incorporates a performance library file into the source.

User-written programs call the Performance Manager procedures using explicit procedure calls, as defined by the syntax of the source language. When you call a Performance Manager procedure, you must supply one argument to the procedure. The argument is the address of the data structure containing all the information needed by the Performance Manager procedure. This location is referred to as the *context block*. When a Performance Manager procedure completes execution, it returns status codes and control to your program. Your program should examine the value of the returned status codes to determine the success or failure of the procedure, and then proceed accordingly.

Related Documentation

The CA guides for CA Performance Management for OpenVMS include the following:

- *Installation Guide*
- *Performance Manager Administrator Guide*
- *Performance Agent Administrator Guide*
- *Release Summary*

The CA guide for common OpenVMS services is:

- *CA Common Services for OpenVMS Integration Guide*

The following HP OpenVMS documents may be useful to readers of this guide:

- *OpenVMS Calling Standard*—Includes information about procedure calling and condition handling
- *OpenVMS Librarian Utility Manual*—Includes information about how to create a library and insert a module

Chapter 2: Performance Manager APIs

This chapter contains the following information about the Performance Manager API:

- Description of the context block for passing information
- Library modules

The sections in this chapter contain descriptions for the Performance Manager API functions, but not the lower-level functions that are called by these supported functions. Your programs should not need to call the lower-level functions directly.

The three Performance Manager API functions follow:

- PSPA$OPEN_CONTEXT defines the view of Performance Manager daily information to be accessed.
- PSPA$READ_CONTEXT sequentially reads daily Performance Manager data records.
- PSPA$CLOSE_CONTEXT closes a previously opened context block.

The address of the context block is passed to the routine as an argument. However, you must allocate memory for the context block before calling the routines. The symbol CTX$S_CONTEXT defines the total length (in bytes) of the context block. The user-written program must insert the appropriate information in the context block prior to the Performance Manager procedure call.

This section contains the following topics:

Data Type Identifiers (see page 9)
Sample Code (see page 10)
PSPA$OPEN_CONTEXT (see page 12)
PSPA$READ_CONTEXT (see page 37)
PSPA$CLOSE_CONTEXT (see page 53)

Data Type Identifiers

The following table lists the data type identifiers used in the field names. For example, the L following the dollar sign in CTX$L_STATUS indicates this field is a longword integer.

Letter	Data Type or Use
A	Address
B	Byte integer
F	Single precision floating
L	Longword integer
M	Field mask
Q	Quadword integer
S	Field size
T	Text (character) string
V	Bit-field position
W	Word integer

Sample Code

This sample code fragment shows how to set up and call the context block and call the PSPA$OPEN_CONTEXT routine. The following HP C sample code is for Context and Collection Definition Blocks:

```
struct CONTEXT ctx_rec;
struct PSPACDSTRUC cd_rec;
.
.
.

status = setup_ctxt(ctx_rec, cd_rec);
.
.
.

ctxt->CTX$L_ARGCNT = sizeof(struct CONTEXT);
ctxt->CTX$r_conio_flags.CTX$r_conio.CTX$V_NETWORK = 1;
ctxt->CTX$r_conio_flags.CTX$r_conio.CTX$V_REALTIME = 1;
ctxt->CTX$A_ASTRTE = &NetAst;
ctxt->CTX$L_ASTARG = ctxt;
.
.
```

```
.

cd->CTXCD$L_RECD_TYPE = 21;
cd->CTXCD$L_RECD_LENGTH = sizeof(struct PSPACDSTRUC);
cd->CTXCD$L_DECPS_VERSION = 0x01010028;
cd->CTXCD$L_INTERVAL = interval;
cd->CTXCD$L_DATA_ACCESS = CTXCD$K_NETWORK;
```

PSPA$OPEN_CONTEXT

PSPA$OPEN_CONTEXT defines the view of the Performance Manager daily information to be accessed. The view is defined by specifying a node name and various day and time inputs to the procedure. PSPA$OPEN_CONTEXT allocates memory that is used by subsequent calls to PSPA$READ_CONTEXT. The memory is deallocated when the context block is closed with a call to PSPA$CLOSE_CONTEXT.

Note: Up to 127 contexts can be opened concurrently. If necessary, the user memory quota can be increased to accommodate a large number of open contexts.

Argument

The argument passed to the routine has the following format:

`PSPA$OPEN_CONTEXT` *`context-block-address`*

context-block-address

These variables define the address of a structure that is used to pass the information, which is listed in the following table, to the procedure:

OpenVMS usage	Address
Type	unsigned longword
access	read-only
mechanism	by reference

Context Block Fields

Each of the following sections describes one context block field that supplies the argument with its sub-argument.

CTX$L_ARGCNT

CTX$L_ARGCNT is the total number of bytes in the context block. Initialize this field with the constant CTX$K_CTXLEN or CTX$S_CONTEXT. Because the value of this constant changes with different versions of the API, you must properly initialize this field.

OpenVMS usage	longword_unsigned
type	unsigned longword

access	read-only
mechanism	by value

CTX$L_STATUS

CTX$L_STATUS is the returned completion status of the procedure. Check this field after each call to any procedure. If an error is signaled, CTX$L_OUR_ERR contains an additional error message code. Also, CTX$L_RMS_ERR might contain an error code.

OpenVMS usage	cond_value
type	unsigned longword
access	write only
mechanism	by value

The following table is for a description of CTX$L_STATUS return status codes:

Status Code	**Description**
PSPA$_NORMAL	Normal successful completion.
PSPA$_ERROR	An error occurred.

CTX$L_CONDS

CTX$L_CONDS is a bit mask that indicates the condition of the context.

OpenVMS usage	mask_longword
type	unsigned longword
access	read/write
mechanism	by value

The CTX$L_CONDS section contains flag bits. The following table describes the flag bits:

Flag-Bit	**Description**
CTX$M_REALTIME (read-only)	Indicates that you have supplied an AST routine to be executed when more real-time data is available (CTX$A_ASTRTE and CTX$L_ASTARG).

Flag-Bit	**Description**
CTX$M_NETWORK (read-only)	Activates a remote data collection process and creates a DECnet link over which to receive data.
CTX$M_NETFILE (read-only)	Activates a data collection process on a node in the cluster and accesses data from the disk file local to the cluster or LAVC that the process creates or updates, and maintains a DECnet link with the data collection process to regulate its data measurements.
CTX$M_DSKFILE (read-only)	Accesses data from a diskfile local to the cluster or LAVC.
CTX$M_FIDDLE (read-only)	Causes the string representation of the FID or the string representation of Non-Virtual QIO to be specified if the name FIL_A_FILE, in a hot file record, is blank (see PSPA$OPEN_CONTEXT).
CTX$M_DNS	Translates cluster node names to DECnet Phase V full names when establishing a network connection for real-time data transport. Define the name translation file name with the PSPA$DNS_NAMES logical in the Process logical name table. See the *Performance Manager Administrator Guide* for more information on this logical name and the translation file.

There are four valid modes of data access. You must use one of the following valid combinations for the flag bits:

- Non-real-time access reads performance data from a file.

  ```
  CTX$M_NETWORK = off CTX$M_DSKFILE = on
  CTX$M_NETFILE = off
  CTX$M_REALTIME = off
  ```

- Real-time disk file access uses data currently being collected by an already existing data collector in a real-time fashion. The file accessed must be for a node in the cluster.

  ```
  CTX$M_NETWORK = off CTX$M_REALTIME = on
  CTX$M_NETFILE = off CTX$M_DSKFILE = on
  ```

- Real-time network access starts a remote real-time data collector, and the data it collects is transmitted back through DECnet in a real-time fashion.

  ```
  CTX$M_DSKFILE = off CTX$M_REALTIME = on
  CTX$M_NETFILE = off CTX$M_NETWORK = on
  ```

- Real-time Netfile access starts a remote real-time data collector, and the collector data is written to a file. Real-time access to the data file is established in a real-time fashion. The remote node must be in the cluster.

  ```
  CTX$M_DSKFILE = off CTX$M_REALTIME = on
  CTX$M_NETWORK = off CTX$M_NETFILE = on
  ```

CTX$L_CONTEXT

CTX$L_CONTEXT is the number assigned to this context by the Performance Manager procedure. The Performance Manager uses the context number as input in subsequent calls to PSPA$READ_CONTEXT and PSPA$CLOSE_CONTEXT. This field is maintained by the performance software.

OpenVMS usage	longword
type	unsigned longword
access	write only
mechanism	by value

CTX$B_NODELEN

CTX$B_NODELEN is the length of the node name text string CTX$T_NODENAME.

OpenVMS usage	byte
type	byte (unsigned)
access	read-only
mechanism	by value

CTX$T_NODENAME

CTX$T_NODENAME is the name of the node whose information you want to access.

OpenVMS usage	char_string
type	character string
access	read-only
mechanism	by value

CTX$L_START

CTX$L_START is the start date and time of the context in 64-bit system time format. It is the low-order 32-bits of the start date and time. This longword and CTX$L_START2 specify the entire start date and time.

OpenVMS usage	longword
type	unsigned longword
access	read-only
mechanism	by value

CTX$L_START2

CTX$L_START2 is the start date and time of the context in 64-bit system time format. It is the high-order 32-bits of the start date and time. This longword and CTX$L_START specify the entire start date and time.

OpenVMS usage	longword
type	unsigned longword
access	read-only
mechanism	by value

CTX$L_STOP

CTX$L_STOP is the stop date and time of the context in 64-bit system time format. It is the low-order 32-bits of the stop date and time. This longword and CTX$L_STOP2 specify the entire stop date and time.

OpenVMS usage	longword
type	unsigned longword
access	read-only
mechanism	by value

CTX$L_STOP2

CTX$L_STOP2 is the stop date and time of the context in 64-bit system time format. It is the high-order 32-bits of the stop date and time. This longword and CTX$L_STOP specify the entire stop date and time.

OpenVMS usage	longword
type	unsigned longword
access	read-only
mechanism	by value

CTX$T_SCHED_SUN

CTX$T_SCHED_SUN is a bit vector indicating which hour time periods for Sundays are to be read in the time range defined by CTX$L_START and CTX$L_STOP. For example, to read Performance Manager data between 2 p.m. and 4 p.m. for every Sunday and Wednesday, you must set bits 14 and 15 in the CTX$T_SCHED_SUN and CTX$T_SCHED_WED bit vectors.

OpenVMS usage	bit_vector
type	character
access	read-only
mechanism	by value

CTX$T_SCHED_MON

CTX$T_SCHED_MON is a bit vector indicating which hour time periods for Mondays are to be read in the time range defined by CTX$L_START and CTX$L_STOP. For example, to read Performance Manager data between 9 a.m. and 11 a.m. for every Monday and Wednesday, you must set bits 9 and 10 in the CTX$T_SCHED_MON and CTX$T_SCHED_WED bit vectors.

OpenVMS usage	bit_vector
type	character
access	read-only
mechanism	by value

CTX$T_SCHED_TUE

CTX$T_SCHED_TUE is a bit vector indicating which hour time periods for Tuesdays are to be read in the time range defined by CTX$L_START and CTX$L_STOP. For example, to read Performance Manager data between 5 p.m. and midnight for every Tuesday and Wednesday, you must set bits 17 through 23 in the CTX$T_SCHED_TUE and CTX$T_SCHED_WED bit vectors.

OpenVMS usage	bit_vector
type	character
access	read-only
mechanism	by value

CTX$T_SCHED_WED

CTX$T_SCHED_WED is a bit vector indicating which hour time periods for Wednesdays are to be read in the time range defined by CTX$L_START and CTX$L_STOP. For example, to read Performance Manager data between midnight and 1 a.m. for every Wednesday, you must set bit 0 in the CTX$T_SCHED_WED bit vector.

OpenVMS usage	bit_vector
type	character
access	read-only
mechanism	by value

CTX$T_SCHED_THU

CTX$T_SCHED_THU is a bit vector indicating which hour time periods for Thursdays are to be read in the time range defined by CTX$L_START and CTX$L_STOP. For example, to read Performance Manager data between 9 a.m. and noon for every Thursday, you must set bits 9 through 11 in the CTX$T_SCHED_THU bit vector.

OpenVMS usage	bit_vector
type	character
access	read-only
mechanism	by value

CTX$T_SCHED_FRI

CTX$T_SCHED_FRI is a bit vector indicating which hour time periods for Fridays are to be read in the time range defined by CTX$L_START and CTX$L_STOP. For example, to read Performance Manager data between 9 a.m. and 5 p.m. for every Friday, you must set bits 9 through 16 in the CTX$T_SCHED_FRI bit vector.

OpenVMS usage	bit_vector
type	character
access	read-only
mechanism	by value

CTX$T_SCHED_SAT

CTX$T_SCHED_SAT is a bit vector indicating which hour time periods for Saturdays are to be read in the time range defined by CTX$L_START and CTX$L_STOP. For example, to read Performance Manager data for the entire day for every Saturday, you must set bits 0 through 23 in the CTX$T_SCHED_SAT bit vector.

OpenVMS usage	bit_vector
type	character
access	read-only
mechanism	by value

CTX$L_OUR_ERR

CTX$L_OUR_ERR is the Performance Manager error message code returned if an error is encountered. This field contains an error message code if CTX$L_STATUS indicates an error.

OpenVMS usage	cond_value
type	unsigned longword
access	write only
mechanism	by value

The following table lists the error codes returned from the previous process:

Error Code	Description
CTX$K_INCOMPAT	Data is incompatible with reader. The application may be using old Performance Manager procedures and libraries. You may have to recompile and relink your application using the latest Performance Manager procedures and libraries.
CTX$K_ERREADFILE	Error accessing a daily data input file. An RMS error has occurred. Check CTX$L_RMS_ERR and CTX$L_RMS_IOSB for specific RMS error information.
CTX$K_NOALLOCVA	Insufficient virtual memory. Increase PGFLQUOTA for the user running the application. Increase the SYSGEN parameter VIRTUALPAGECNT, if it is less than PGFLQUOTA.

CTX$L_RMS_ERR

CTX$L_RMS_ERR is the RMS or OpenVMS error message code returned if an error is encountered within a system routine. This field may not always contain an error message code but should be checked if an error condition is indicated by CTX$L_STATUS:

OpenVMS usage	cond_value
type	unsigned longword
access	write only
mechanism	by value

CTX$L_RMS_IOSB

CTX$L_RMS_IOSB is a device-specific error information returned when an RMS error is encountered. This field may not always contain information but should be checked if an error condition is indicated by CTX$L_RMS_ERR.

OpenVMS usage	cond_value
type	unsigned longword
access	write only
mechanism	by value

CTX$A_ASTRTE

CTX$A_ASTRTE is the address of a user Asynchronous System Trap (AST) routine to be executed when a data collection file has generated more data and made it accessible, either from a disk file or over the DECnet link. This argument is usually used with CTX$L_ASTARG and CTX$V_REALTIME.

OpenVMS usage	address
type	unsigned longword
access	read-only
mechanism	by reference

CTX$L_ASTARG

CTX$L_ASTARG is a user argument passed to the AST routine specified in CTX$A_ASTRTE. This argument is usually used with CTX$A_ASTRTE and CTX$V_REALTIME.

OpenVMS usage	longword
type	unsigned longword
access	read-only
mechanism	by value

CTX$L_COLL_DEF_LEN

CTX$L_COLL_DEF_LEN is the length of the collection definition name, specified by CTX$T_COLL_DEF_NAME.

OpenVMS usage	longword
type	unsigned longword
access	read-only
mechanism	by value

CTX$T_COLL_DEF_NAME

CTX$T_COLL_DEF_NAME is the text string name of the collection definition. The name can be 1 to 15 characters long. For access to the primary collector's data, specify CPD in this field. This field must match the collection name specified in the collection definition pointed to in CTX$A_COLL_DEF_PTER. It is used in identifying collection files.

OpenVMS usage	character
type	character
access	read-only
mechanism	by value

CTX$A_COLL_DEF_PTER

CTX$A_COLL_DEF_PTER is the address of a collection definition that specifies the attributes of the data accessed by the callable interface and, where a data collection process is initiated, is passed to that process. The module $PSPACDDEFS in PSPA$LIB describes the collection definition data structure to be used to construct the data structure that this argument references.

OpenVMS usage	address
type	unsigned longword
access	read-only
mechanism	by value

The collection definition construct is defined by the PSPADCSTRUC data structure definition in the PSPA$LIB library file.

CTXCD$L_RECD_TYPE

CTXCD$L_RECD_TYPE is the Collection Definition subrecord type (Decimal 21).

OpenVMS usage	longword
type	unsigned longword
access	read-only
mechanism	by value

CTXCD$L_RECD_LENGTH

CTXCD$L_RECD_LENGTH is the Collection Definition subrecord length CTXCD$S_PSPACDSTRUC.

OpenVMS usage	longword
type	unsigned longword
access	read-only
mechanism	by value

CTXCD$L_DECPS_VERSION

CTXCD$L_DECPS_VERSION is the Performance Manager version number (Hex 01010028).

OpenVMS usage	longword
type	unsigned longword
access	read-only
mechanism	by value

CTXCD$L_NODENAME_LEN

CTXCD$L_NODENAME_LEN is the length of the nodename text string CTXCD$T_NODENAME.

OpenVMS usage	longword
type	unsigned longword
access	read-only
mechanism	by value

CTXCD$T_NODENAME

CTXCD$T_NODENAME is the nodename text string of the node whose data is to be accessed or created.

OpenVMS usage	character
type	character
access	read-only
mechanism	by value

CTXCD$L_COLL_DEF_LEN

CTXCD$L_COLL_DEF_LEN is the length of the collection definition name, specified by CTX$T_COLL_DEF_NAME.

OpenVMS usage	longword
type	unsigned longword
access	read-only
mechanism	by value

CTXCD$T_COLL_DEF_NAME

CTXCD$T_COLL_DEF_NAME is the text string name of the collection definition. The name may be 1 to 15 characters in length.

OpenVMS usage	character
type	character
access	read-only
mechanism	by value

CTXCD$A_FLINK

CTXCD$A_FLINK is reserved for the use of CA.

CTXCD$L_ITVL_QUOTA

CTXCD$L_ITVL_QUOTA is the flow control between an initiated data collection process and the callable interface over a DECnet link. The flow control is in terms of the number of intervals of data that can be queued at the receiving end of the link at the callable interface.

A value of 0 (zero) directs the data collection process to wait until the callable interface is done with all its data and sends a message that it wants more. In the meantime, the data collection process suspends measurements and there is no build up of data buffers, transmission of data, or CPU expenditure.

A value of 1 or more directs the data collection process to try to maintain a queue of that number of intervals of data at the callable interface. Fixed length intervals of contiguous time are delivered until the queue limit is exceeded.

OpenVMS usage	longword
type	unsigned longword
access	read-only
mechanism	by value

CTXCD$L_DATA_ACCESS

CTXCD$L_DATA_ACCESS is a directive to an initiated data collection process specifying the type of data to produce.

OpenVMS usage	mask_longword
type	unsigned longword
access	read-only
mechanism	by value

CTXCD$V_MAINMAN

CTXCD$V_MAINMAN is reserved for the use of CA.

CTXCD$V_NETWORK

CTXCD$V_NETWORK is a directive to initiate a remote data collection process, establish a DECnet link, and deliver the performance data over this link. Flow control is maintained by messages between the initiated data collection process and this instance of the callable interface.

CTXCD$V_NETFILE

CTXCD$V_NETFILE is a directive to initiate a remote data collection process, establish a DECnet link, and record the performance data in a disk file that the callable interface then accesses. Flow control is maintained by messages between the initiated data collection process and this instance of the callable interface.

CTXCD$V_DSKFILE

CTXCD$V_DSKFILE is a directive to initiate a local detached data collection process with no explicit dependency on any instance of the callable interface.

CTXCD$L_INTERVAL

CTXCD$L_INTERVAL is the length of the data collection measurement interval in seconds. Each interval record is of this duration.

OpenVMS usage	longword
type	unsigned longword
access	read-only
mechanism	by value

CTXCD$L_CLASSES

CTXCD$L_CLASSES is a mask specifying which subrecord types to generate.

OpenVMS usage	mask_longword
type	unsigned longword
access	read-only
mechanism	by value

CTXCD$V_CLASS_STATUS

CTXCD$V_CLASS_STATUS is a flag that indicates a return status indicator for any routine that references this argument. The status is indicated by:

- 0=success
- 1=failure

CTXCD$V_PROCESS

CTXCD$V_PROCESS is a flag directive to generate process subrecords.

CTXCD$V_IMAGE

CTXCD$V_IMAGE is reserved for the use of CA.

CTXCD$V_DISK

CTXCD$V_DISK is a flag directive to generate disk subrecords.

CTXCD$V_TAPE

CTXCD$V_TAPE is a flag directive to generate tape subrecords.

CTXCD$V_PARAMS

CTXCD$V_PARAMS is a flag directive to generate parameter subrecords.

CTXCD$V_SYSTEM_METRICS

CTXCD$V_SYSTEM_METRICS is a flag directive to generate system metric subrecords.

CTXCD$V_TERMINAL

CTXCD$V_TERMINAL is a flag directive to generate terminal controller subrecords.

CTXCD$V_CONFIGURATION

CTXCD$V_CONFIGURATION is a flag directive to generate SCS configuration subrecords.

CTXCD$V_CPU

CTXCD$V_CPU is a flag directive to generate CPU subrecords.

CTXCD$V_HOT_FILE

CTXCD$V_HOT_FILE is reserved for the use of CA.

CTXCD$V_HOT_LOCKS

CTXCD$V_HOT_LOCKS is reserved for the use of CA.

CTXCD$V_DECTRACE

CTXCD$V_DECTRACE is reserved for the use of CA.

CTXCD$V_FULL_STRINGS

CTXCD$V_FULL_STRINGS is a flag directive to collect the full image specification including disk, directory, and image name.

CTXCD$V_CLASS_SPARES

CTXCD$V_CLASS_SPARES is reserved for the use of CA.

CTXCD$L_CLASSES2

CTXCD$L_CLASSES2 is reserved for the use of CA.

CTXCD$L_CLASSES3

CTXCD$L_CLASSES3 is reserved for the use of CA.

CTXCD$L_CLASSES4

CTXCD$L_CLASSES4 is reserved for the use of CA.

CTXCD$L_HOTFILE_QUEUE

CTXCD$L_HOTFILE_QUEUE is the hot file disk queue length threshold controlling the measurement and recording of hot file data in the main data collection process, based on the CPD collection definition.

OpenVMS usage	longword
type	unsigned longword
access	read-only
mechanism	by value

CTXCD$L_START_TIME

CTXCD$L_START_TIME is the low order of a 64-bit system time specifying the start of the timeframe during which the data collection process image is run.

OpenVMS usage	longword
type	unsigned longword
access	read-only
mechanism	by value

CTXCD$L_START_TIME2

CTXCD$L_START_TIME2 is the high order of a 64-bit system time specifying the start of the timeframe during which the data collection process image is run.

OpenVMS usage	longword
type	unsigned longword
access	read-only
mechanism	by value

CTXCD$L_END_TIME

CTXCD$L_END_TIME is the low order of a 64-bit system time specifying the end of the timeframe during which the data collection process image is run.

OpenVMS usage	longword
type	unsigned longword
access	read-only
mechanism	by value

CTXCD$L_END_TIME2

CTXCD$L_END_TIME2 is the high order of a 64-bit system time specifying the end of the time frame during which the data collection process image is run.

OpenVMS usage	longword
type	unsigned longword
access	read-only
mechanism	by value

CTXCD$L_WS_QUOTA

CTXCD$L_WS_QUOTA is the Working Set Quota of the data collection process.

OpenVMS usage	longword
type	unsigned longword
access	read-only
mechanism	by value

CTXCD$L_WS_EXTENT

CTXCD$L_WS_EXTENT is the Working Set Extent of the data collection process.

OpenVMS usage	longword
type	unsigned longword
access	read-only
mechanism	by value

CTXCD$L_CPU_THRESHOLD

CTXCD$L_CPU_THRESHOLD is reserved for the use of CA.

CTXCD$L_RESOURCE_NAME_LEN

CTXCD$L_RESOURCE_NAME_LEN is the length of the resource name associated with this collection definition.

OpenVMS usage	longword
type	unsigned longword
access	read-only
mechanism	by value

CTXCD$T_RESOURCE_NAME

CTXCD$T_RESOURCE_NAMEis the resource name associated with this collection definition.

OpenVMS usage	character
type	character
access	read-only
mechanism	by value

CTXCD$L_PATH_LEN

CTXCD$L_PATH_LEN is the length of the path specification associated with this collection definition.

OpenVMS usage	longword
type	unsigned longword
access	read-only
mechanism	by value

CTXCD$T_PATH_SPEC

CTXCD$T_PATH_SPEC is the name of the path specification associated with this collection definition. The path specification includes both the disk and directory where the data file is recorded.

OpenVMS usage	character
type	character
access	read-only
mechanism	by value

CTXCD$L_SCHED_MON

CTXCD$L_SCHED_MON is a bit vector indicating which hour time periods for Mondays are to be read in the time range defined by CTXCD$L_START and CTXCD$L_STOP. For example, to read Performance Manager data between 9 a.m. and 11 a.m. for every Monday and Wednesday, you must set bits 9 and 10 in the CTXCD$L_SCHED_MON and CTXCD$L_SCHED_WED bit vectors.

OpenVMS usage	bit_vector
type	character
access	read-only
mechanism	by value

CTXCD$L_SCHED_TUE

CTXCD$L_SCHED_TUE is a bit vector indicating which hour time periods for Tuesdays are to be read in the time range defined by CTXCD$L_START and CTXCD$L_STOP. For example, to read Performance Manager data between 5 p.m. and midnight for every Tuesday and Wednesday, you must set bits 17 through 23 in the CTXCD$L_SCHED_TUE and CTXCD$L_SCHED_WED bit vectors.

OpenVMS usage	bit_vector
type	character
access	read-only
mechanism	by value

CTXCD$L_SCHED_WED

CTXCD$L_SCHED_WED is a bit vector indicating which hour time periods for Wednesdays are to be read in the time range defined by CTXCD$L_START and CTXCD$L_STOP. For example, to read Performance Manager data between midnight and 1 a.m. for every Wednesday, you must set bit 0 in the CTXCD$L_SCHED_WED bit vector.

OpenVMS usage	bit_vector
type	character
access	read-only
mechanism	by value

CTXCD$L_SCHED_THU

CTXCD$L_SCHED_THU is a bit vector indicating which hour time periods for Thursdays are to be read in the time range defined by CTXCD$L_START and CTXCD$L_STOP. For example, to read Performance Manager data between 9 a.m. and noon for every Thursday, you must set bits 9 through 11 in the CTXCD$L_SCHED_THU bit vector.

OpenVMS usage	bit_vector
type	character
access	read-only
mechanism	by value

CTXCD$L_SCHED_FRI

CTXCD$L_SCHED_FRI is a bit vector indicating which hour time periods for Fridays are to be read in the time range defined by CTXCD$L_START and CTXCD$L_STOP. For example, to read Performance Manager data between 9 a.m. and 5 p.m. for every Friday, you must set bits 9 through 16 in the CTXCD$L_SCHED_FRI bit vector.

OpenVMS usage	bit_vector
type	character
access	read-only
mechanism	by value

CTXCD$L_SCHED_SAT

CTXCD$L_SCHED_SAT is a bit vector indicating which hour time periods for Saturdays are to be read in the time range defined by CTXCD$L_START and CTXCD$L_STOP. For example, to read Performance Manager data for the entire day for every Saturday, you must set bits 0 through 23 in the CTXCD$L_SCHED_SAT bit vector.

OpenVMS usage	bit_vector
type	character
access	read-only
mechanism	by value

CTXCD$L_SCHED_SUN

CTXCD$L_SCHED_SUN is a bit vector indicating which hour time periods for Sundays are to be read in the time range defined by CTXCD$L_START and CTXCD$L_STOP. For example, to read Performance Manager data between 2 p.m. and 4 p.m. for every Sunday and Wednesday, you must set bits 14 and 15 in the CTXCD$L_SCHED_SUN and CTXCD$L_SCHED_WED bit vectors.

OpenVMS usage	bit_vector
type	character
access	read-only
mechanism	by value

CTXCD$L_MAXDEVCNT

CTXCD$L_MAXDEVCNT is reserved for the use of CA.

CTXCD$L_HOTFILE_MAX

CTXCD$L_HOTFILE_MAX is reserved for the use of CA.

CTXCD$L_HOTFILE_LIFE

CTXCD$L_HOTFILE_LIFE is reserved for the use of CA.

CTXCD$L_RETENTION

CTXCD$L_RETENTION is the number of days to retain data files recorded and named according to this collection definition.

OpenVMS usage	longword
type	unsigned longword
access	read-only
mechanism	by value

CTXCD$L_DISKSPACE

CTXCD$L_DISKSPACE is the number of free blocks required to be available on a disk to record data on that disk.

OpenVMS usage	longword
type	unsigned longword
access	read-only
mechanism	by value

CTXCD$L_DISKINCEXC

CTXCD$L_DISKINCEXC is reserved for the use of CA.

CTXCD$L_PROCINCEXC

CTXCD$L_PROCINCEXC is reserved for the use of CA.

CTX$A_DATE_LIST

CTX$A_DATE_LIST is a pointer to a linked list of standard stop times of subsequent periods of time to be accessed after the original start and stop times as specified by CTX$L_START and CTX$L_STOP are accessed. Each entry in the list consists of a longword pointer to the next entry where a zero terminates the list, followed by a start time in 64-bit system time, followed by a stop time in 64-bit system time. All periods of time must be chronologically forward. Data structure $PSPADATESTRUC in PSPA$LIB is provided.

OpenVMS usage	address
type	unsigned longword
access	read-only
mechanism	by reference

The following context block is a description of the data structure that is referenced by the context block argument CTX$A_DATE_LIST. The construct is defined by the data structure PSPADATESTRUC in the PSPA$LIB library file.

CTXDATE$A_FLINK

CTXDATE$A_FLINK is a pointer to the next entry in the date list. It is a zero terminated list.

OpenVMS usage	address
type	unsigned longword
access	read-only
mechanism	by reference

CTXDATE$L_FROM_TIME

CTXDATE$L_FROM_TIME is the low order of a 64-bit system time that specifies the start date and time of the next time frame of data to access.

OpenVMS usage	longword
type	unsigned longword
access	read-only
mechanism	by value

CTXDATE$L_FROM_TIME2

CTXDATE$L_FROM_TIME2 is the high order of a 64-bit system time that specifies the start date and time of the next time frame of data to access.

OpenVMS usage	longword
type	unsigned longword
access	read-only
mechanism	by value

CTXDATE$L_TO_TIME

CTXDATE$L_TO_TIME is the low order of a 64-bit system time that specifies the end date and time of the next time frame of data to access.

OpenVMS usage	longword
type	unsigned longword
access	read-only
mechanism	by value

CTXDATE$L_TO_TIME2

CTXDATE$L_TO_TIME2 is the high order of a 64-bit system time that specifies the end date and time of the next time frame of data to access.

OpenVMS usage	longword
type	unsigned longword
access	read-only
mechanism	by value

Possible Return Values

The following values might be returned when the routine closes.

OpenVMS usage	cond_value
type	unsigned longword
access	write only
mechanism	as a function value and in CTX$L_STATUS; in MACRO-32, this return value is in R0

PSPA$READ_CONTEXT

PSPA$READ_CONTEXT sequentially reads daily Performance Manager data records. The PSPA$READ_CONTEXT routine sequentially reads Performance Manager daily data for the time periods specified in the PSPA$OPEN_CONTEXT call. Memory is allocated by the procedure and all of the records associated with a collection interval are returned in this memory.

Argument

The argument passed to the routine follows this format:

```
PSPA$OPEN_CONTEXT context-block-address
```

context-block-address

These variables define the address of a structure that is used to pass the information, listed in the following table, to the procedure:

OpenVMS usage	Address
Type	unsigned longword
access	read-only
mechanism	by reference

Context Block Fields

Each of the following sections describes one context block field that supplies the argument with its sub-argument.

CTX$L_ARGCNT

CTX$L_ARGCNT is the total number of bytes in the context block. Initialize this field with the constant CTX$K_CTXLEN or CTX$S_CONTEXT. Because the value of this constant changes with different versions of the callable interface, you must properly initialize this field.

OpenVMS usage	longword_unsigned
type	unsigned longword
access	read-only
mechanism	by value

CTX$L_STATUS

CTX$L_STATUS is the returned completion status of the procedure. Check this field after each call to the procedure. If an error is signaled, CTX$L_OUR_ERR contains an additional error message code. Also, CTX$L_RMS_ERR may contain an error code.

OpenVMS usage	cond_value
type	unsigned longword

access	write only
mechanism	by value

The following table lists the return status codes from the previous process:

Status Code	Description
PSPA$_NORMAL	Normal successful completion.
PSPA$_ERROR	An error occurred.
PSPA$_NOMORE	No data available for specified context.

Note: If CTX$M_PARTIAL is set on return in CTX$L_CONDS with PSPA$_NOMORE in CTX$L_STATUS, the CIFDE is waiting for more data to become available.

CTX$L_CONDS

CTX$L_CONDS is the bit mask indicating the condition of the context.

OpenVMS usage	mask_longword
type	unsigned longword
access	write only
mechanism	by value

The following table lists the condition mask symbols and descriptions for the previous process:

Symbol	Description
CTX$M_NOMORE	No more data
CTX$M_HOLE	Hole in data
CTX$M_LOST	Unintelligible data
CTX$M_IOERR	Error accessing file
CTX$M_PARTIAL	At today's partial record
CTX$M_INCOMPAT	Incompatible data

CTX$L_CONTEXT

CTX$L_CONTEXT is associated with this context by the PSPA$OPEN_CONTEXT.

OpenVMS usage	longword
type	unsigned longword
access	read-only
mechanism	by value

CTX$L_HOLE_BGN

CTX$L_HOLE_BGN is the beginning date and time of the missing data within the context in 64-bit system time format. It is the low-order 32-bits of the missing data date and time. This longword and CTX$L_HOLE_BGN2 specify the begin date and time of the missing data.

OpenVMS usage	longword
type	unsigned longword
access	write only
mechanism	by value

CTX$L_HOLE_BGN2

CTX$L_HOLE_BGN2 is the date and time of the missing data within the context in 64-bit system time format. It is the high-order 32-bits of the missing data date and time. This longword and CTX$L_HOLE_BGN specify the begin date and time of the missing data.

OpenVMS usage	longword
type	unsigned longword
access	write only
mechanism	by value

CTX$L_HOLE_END

CTX$L_HOLE_END is the end date and time of the missing data within the context in 64-bit system time format. It is the low-order 32-bits of the missing data date and time. This longword and CTX$L_HOLE_END2 specify the end date and time of the missing data.

OpenVMS usage	longword
type	unsigned longword
access	write only
mechanism	by value

CTX$L_HOLE_END2

CTX$L_HOLE_END2 is the end date and time of the missing data within the context in 64-bit system time format. It is the high-order 32-bits of the missing data date and time. This longword and CTX$L_HOLE_END specify the end date and time of the missing data.

OpenVMS usage	longword
type	unsigned longword
access	write only
mechanism	by value

CTX$L_OUR_ERR

CTX$L_OUR_ERR is the Performance Manager error message code that is returned if an error is encountered. This field contains an error message code if CTX$L_STATUS indicates an error.

OpenVMS usage	cond_value
type	unsigned longword
access	write only
mechanism	by value

The following table lists the error codes from the previous process:

Error Code	Description
CTX$K_INCOMPAT	Data is incompatible with reader. The application may be using old Performance Manager procedures and libraries. You may have to recompile and relink your application using the latest Performance Manager procedures and libraries.
CTX$K_ERREADFILE	Error accessing a daily data input file. An RMS error has occurred. Check CTX$L_RMS_ERR and CTX$L_RMS_IOSB for specific RMS error information.
CTX$K_NOALLOCVA	Insufficient virtual memory. Increase PGFLQUOTA for the user running the application.

CTX$L_RMS_ERR

The RMS or OpenVMS error message code CTX$L_RMS_ERR is returned if an error is encountered in a system routine. This field may not always contain an error message code but should be checked if CTX$L_STATUS indicates an error condition.

OpenVMS usage	cond_value
type	unsigned longword
access	write only
mechanism	by value

CTX$L_RMS_IOSB

CTX$L_RMS_IOSB is device-specific error information returned when an RMS error is encountered. This field may not always contain information but should be checked if CTX$L_RMS_ERR indicates an error condition.

OpenVMS usage	cond_value
type	unsigned longword
access	write only
mechanism	by value

CTX$A_TIM_REC

CTX$A_TIM_REC is the address of a buffer that contains the time record for the current interval.

OpenVMS usage	address
type	unsigned longword
access	write only
mechanism	by reference

CTX$A_MET_REC

CTX$A_MET_REC is the address of a buffer that contains the metrics record for the current interval.

OpenVMS usage	address
type	unsigned longword
access	write only
mechanism	by reference

CTX$A_PAR_REC

CTX$A_PAR_REC The address of a buffer that contains the parameter record for the current interval.

OpenVMS usage	address
type	unsigned longword
access	write only
mechanism	by reference

CTX$A_PRO_REC

CTX$A_PRO_REC is the address of a buffer that contains the process records for the current interval.

OpenVMS usage	address
type	unsigned longword
access	write only

mechanism	by reference

CTX$A_DEV_REC

CTX$A_DEV_REC is the address of a buffer that contains the device records for the current interval.

OpenVMS usage	address
type	unsigned longword
access	write only
mechanism	by reference

CTX$A_COM_REC

CTX$A_COM_REC is the address of a buffer that contains the communications records for the current interval.

OpenVMS usage	address
type	unsigned longword
access	write only
mechanism	by reference

CTX$A_MAG_REC

CTX$A_MAG_REC is the address of a buffer that contains the tape records for the current interval.

OpenVMS usage	address
type	unsigned longword
access	write only
mechanism	by reference

CTX$A_CFG_REC

CTX$A_CFG_REC is the address of a buffer that contains the configuration records for the current interval.

OpenVMS usage	address

type	unsigned longword
access	write only
mechanism	by reference

CTX$A_CPU_REC

CTX$A_CPU_REC is the address of a buffer that contains the CPU records for the current interval.

OpenVMS usage	address
type	unsigned longword
access	write only
mechanism	by reference

CTX$A_FIL_REC

CTX$A_FIL_REC is the address of a buffer that contains the hot file records for the current interval.

OpenVMS usage	address
type	unsigned longword
access	write only
mechanism	by reference

CTX$L_TIM_CNT

CTX$L_TIM_CNT is the number of time records in the buffer.

OpenVMS usage	longword
type	unsigned longword
access	write only
mechanism	by value

CTX$L_MET_CNT

CTX$L_MET_CNT is the number of metrics records in the buffer.

OpenVMS usage	longword
type	unsigned longword
access	write only
mechanism	by value

CTX$L_PAR_CNT

CTX$L_PAR_CNT is the number of parameters records in the buffer.

OpenVMS usage	longword
type	unsigned longword
access	write only
mechanism	by value

CTX$L_PRO_CNT

CTX$L_PRO_CNT is the number of process records in the buffer.

OpenVMS usage	longword
type	unsigned longword
access	write only
mechanism	by value

CTX$L_DEV_CNT

CTX$L_DEV_CNT is the number of device records in the buffer.

OpenVMS usage	longword
type	unsigned longword
access	write only
mechanism	by value

CTX$L_COM_CNT

CTX$L_COM_CNT is the number of communications records in the buffer.

OpenVMS usage	longword
type	unsigned longword
access	write only
mechanism	by value

CTX$L_MAG_CNT

CTX$L_MAG_CNT is the number of tape records in the buffer.

OpenVMS usage	longword
type	unsigned longword
access	write only
mechanism	by value

CTX$L_CFG_CNT

CTX$L_CFG_CNT is the number of configuration records in the buffer.

OpenVMS usage	longword
type	unsigned longword
access	write only
mechanism	by value

CTX$L_CPU_CNT

CTX$L_CPU_CNT is the number of CPU records in the buffer.

OpenVMS usage	longword
type	unsigned longword
access	write only
mechanism	by value

CTX$L_FIL_CNT

CTX$L_FIL_CNT is the number of hot file records in the buffer.

OpenVMS usage	longword
type	unsigned longword
access	write only
mechanism	by value

CTX$L_TIM_OVF

CTX$L_TIM_OVF is the number of time records that could not be loaded in the buffer because of memory limitations.

OpenVMS usage	longword
type	unsigned longword
access	write only
mechanism	by value

CTX$L_MET_OVF

CTX$L_MET_OVF is the number of metrics records that could not be loaded in the buffer because of memory limitations.

OpenVMS usage	longword
type	unsigned longword
access	write only
mechanism	by value

CTX$L_PAR_OVF

CTX$L_PAR_OVF is the number of parameter records that could not be loaded in the buffer because of memory limitations.

OpenVMS usage	longword
type	unsigned longword
access	write only
mechanism	by value

CTX$L_PRO_OVF

CTX$L_PRO_OVF is the number of process records that could not be loaded in the buffer because of memory limitations.

OpenVMS usage	longword
type	unsigned longword
access	write only
mechanism	by value

CTX$L_DEV_OVF

CTX$L_DEV_OVF is the number of device records that could not be loaded in the buffer because of memory limitations.

OpenVMS usage	longword
type	unsigned longword
access	write only
mechanism	by value

CTX$L_COM_OVF

CTX$L_COM_OVF is the number of communications records that could not be loaded in the buffer because of memory limitations.

OpenVMS usage	longword
type	unsigned longword
access	write only
mechanism	by value

CTX$L_MAG_OVF

CTX$L_MAG_OVF is the number of tape records that could not be loaded in the buffer because of memory limitations.

OpenVMS usage	longword
type	unsigned longword
access	write only
mechanism	by value

CTX$L_CFG_OVF

CTX$L_CFG_OVF is the number of configuration records that could not be loaded in the buffer because of memory limitations.

OpenVMS usage	longword
type	unsigned longword
access	write only
mechanism	by value

CTX$L_CPU_OVF

CTX$L_CPU_OVF is the number of CPU records that could not be loaded in the buffer because of memory limitations.

OpenVMS usage	longword
type	unsigned longword
access	write only
mechanism	by value

CTX$L_FIL_OVF

CTX$L_FIL_OVF is the number of hot file records that could not be loaded in the buffer because of memory limitations.

OpenVMS usage	longword
type	unsigned longword
access	write only
mechanism	by value

CTX$B_FILE_LEN

CTX$B_FILE_LEN is the length of the file specification pointed to by CTX$A_FILE_PTR.

OpenVMS usage	byte
type	byte (unsigned)
access	write only
mechanism	by value

CTX$A_FILE_PTR

CTX$A_FILE_PTR is the address of the file specification that contains the Performance Manager data currently being read by this context.

OpenVMS usage	address
type	unsigned longword
access	write only
mechanism	by reference

CTX$L_DATA_VERSION

CTX$L_DATA_VERSION is the version of the Performance Agent that recorded the data returned by the latest call to CTX$READ_CONTEXT. The version is represented in the hexadecimal format *vvrreeee*, where *vv* is the version, *rr* is the release, and *eeee* is the edit number.

OpenVMS usage	longword
type	unsigned longword
access	write only
mechanism	by value

Possible Return Values

The address of each of the record buffers with a record count is returned to the calling program by the procedure. PSPA$READ_CONTEXT returns a count in the appropriate overflow field of the number of records that could not be returned because of insufficient memory.

If PSPA$READ_CONTEXT detects missing information while reading the data, the CTX$M_HOLE flag is set in the CTX$L_CONDS bit mask. The time frame for which data is missing is returned in the CTX$L_HOLE_BGN and CTX$L_HOLE_END fields.

PSPA$CLOSE_CONTEXT

PSPA$CLOSE_CONTEXT closes a previously opened context block and releases all memory allocated for the record buffers. The address of the first byte of a structure is used to pass information to the procedure.

Argument

The argument passed to the routine follows this format:

`PSPA$OPEN_CONTEXT` *`context-block-address`*

context-block-address

These variables define the address of a structure that is used to pass to the procedure the information listed in the following table:

OpenVMS usage	Address
Type	unsigned longword
access	read-only
mechanism	by reference

Context Block Fields

Each of the following sections describes one context block field that supplies the argument with its sub-arguments.

CTX$L_ARGCNT

CTX$L_ARGCNT is the total number of bytes in the context block. Initialize this field with the constant CTX$K_CTXLEN or CTX$S_CONTEXT. Because the value of this constant changes with different versions of the callable interface, you must properly initialize this field.

OpenVMS usage	longword_unsigned
type	unsigned longword
access	read-only
mechanism	by value

CTX$L_STATUS

CTX$L_STATUS is the returned completion status of the procedure. This field should be checked after each call to the procedure. If an error is signaled, CTX$L_OUR_ERR contains an additional error message code. Also, CTX$L_RMS_ERR may contain an error code.

OpenVMS usage	cond_value
type	unsigned longword
access	write only
mechanism	by value

Possible Return Values

The following table contains the return status codes and descriptions for PSPA$CLOSE_CONTEXT:

Status Code	**Description**
PSPA$_NORMAL	Normal successful completion.
PSPA$_ERROR	An error occurred.

CTX$L_CONDS

CTX$L_CONDS is a bit mask that indicates the condition of the context.

OpenVMS usage	mask_longword
type	unsigned longword
access	write only
mechanism	by reference

The following table contains a description of the CTX$L_CONDS condition mask symbol:

Symbol	**Description**
CTX$M_IOERR	Error accessing file

CTX$L_CONTEXT

CTX$L_CONTEXT is the number associated with this context by the PSPA$OPEN_CONTEXT.

OpenVMS usage	longword
type	unsigned longword
access	read-only
mechanism	by value

CTX$L_OUR_ERR

CTX$L_OUR_ERR is the Performance Manager error message code that is returned if an error is encountered. This field contains an error message code if CTX$L_STATUS indicates an error.

OpenVMS usage	cond_value
type	unsigned longword
access	write only
mechanism	by value

The following table lists the error codes from the previous process:

Error Code	**Description**
CTX$K_ERREADFILE	Error accessing a daily data input file. An RMS error has occurred. Check CTX$L_RMS_ERR and CTX$L_RMS_IOSB for specific RMS error information.

CTX$L_RMS_ERR

CTX$L_RMS_ERR is the RMS error message code CTX$L_RMS_ERR is returned if an error is encountered in a system routine. This field may not always contain an error message code but should be checked if CTX$L_STATUS indicates an error condition.

OpenVMS usage	cond_value
type	unsigned longword
access	write only

mechanism	by value

CTX$L_RMS_IOSB

CTX$L_RMS_IOSB is the device specific error information returned when an RMS error is encountered. This field might not always contain information but should be checked if CTX$L_RMS_ERR indicates an error condition.

OpenVMS usage	cond_value
type	unsigned longword
access	write only
mechanism	by value

Chapter 3: PSPA Libraries

PSPA libraries support the PSPA$READ.EXE API interface. The language files are *include* or *require* files needed to write the programs that use the API. During installation you can indicate whether you want the PSPA library modules for a particular language placed in the SYS$LIBRARY area on your system. You can select any or all of the library modules in the following table:

Module	Language
PSPA$LIB.BAS	HP Basic
PSPA$LIB.H	HP C
PSPA$LIB.FOR	HP Fortran
PSPA$LIB.MAR	OpenVMS MACRO
PSPA$LIB.PAS	HP Pascal

At installation time, the Performance Manager inserts the PSPA$READ.EXE shareable image into SYS$LIBRARY:IMAGELIB.OLB. This image is used to execute the API procedures. By default, SYS$LIBRARY:IMAGELIB.OLB is searched during the linking process.

This section contains the following topics:

Compile and Link with the PSPA MACRO Library (see page 57)
PSPA Record Field Definitions (see page 58)

Compile and Link with the PSPA MACRO Library

Before compiling your OpenVMS MACRO program for the first time, you must create the PSPA MACRO library and insert the PSPA MACRO library module into the library. You need to do this only once unless the library is deleted.

Use the Librarian Utility to create a library and insert a module. For more information see HP's *OpenVMS Command Definition, Librarian, and Message Utilities Manual*. The following command creates a PSPA MACRO library in your current area:

```
$ LIBRARY/MACRO/CREATE PSPA$LIB.MLB SYS$LIBRARY:PSPA$LIB.MAR
```

When you write a program, for example READPSPA.MAR, that uses the library, you must compile your program with the library file. The following command compiles READPSPA.MAR:

```
$ MACRO READPSPA + PSPA$LIB.MLB/LIB
```

To create an executable image, issue the following command:

```
$ LINK READPSPA
```

A sample OpenVMS MACRO program that uses the Performance Manager procedures is located in the PSPA$EXAMPLES:PSPA$GETDATA.MAR file. For information on accessing the sample program, see the appendix *Performance Manager Sample Programs*.

PSPA Record Field Definitions

The tables in this section list the Performance Manager record fields and their corresponding definitions:

- Configuration Record Field Definitions
- Process Record Field Definitions
- Metrics Record Field Definitions
- CPU Record Field Definitions
- Parameter Record Field Definitions
- Time Record Field Definitions
- Device Field Record Field Definitions
- Tape Record Field Definitions
- Communications Record Field Definitions
- Hot File Record Field Definitions

The symbols listed in the left columns are offsets from the buffer address returned from the Performance Manager procedures such as CTX$A_PRO_REC and CTX$A_MET_REC.

Configuration Record

The following table lists the Configuration Record field definitions:

Field Offset	Definition
CFG_B_TYPE	Record type (9)
CFG_A_NODENAME	Node name ASCID address
CFG_A_PATH	Path ASCID address (example: PAA0)
CFG_L_STATUS	Node status
CFG_V_STATUS_MEMBER	Cluster member
CFG_V_STATUS_HSC	HSC controller
CFG_V_STATUS_VAXNODE	node CPU
CFG_V_STATUS_NI	NI component
CFG_V_STATUS_CI	CI component
CFG_V_STATUS_RF	RF series component
SCS_F_DGSENT	Number of datagrams sent/sec
SCS_F_DGRCVD	Number datagrams received/sec
SCS_F_DGDISCARD	Number datagrams discarded/sec
SCS_F_MSGSENT	Number sequenced messages sent/sec
SCS_F_MSGRCVD	Number sequenced messages received/sec
SCS_F_SNDATS	Number block send datas initiated/sec
SCS_F_KBYTSENT	Number Kilobytes sent using send data/sec
SCS_F_REQDATS	Number block request data initiated
SCS_F_KBYTREQD	Number kilobytes received using request data/sec
SCS_F_KBYTMAPD	Number kilobytes mapped for block transfers/sec
SCS_F_QCR_CNT	Number connections queued for send credit/sec
SCS_F_QBDT_CNT	Number connections queued for buffer descs/sec

Field Offset	Definition
CFG_A_HWNAME	V5.0 node hardware name ASCID address
CFG_L_ADAPTER_ID	CI/NI adapter code
CFG_A_ADAPTER	CI/NI adapter name ASCID addr
CFG_A_SYSAP	Not used

Process Record

The following table lists the Process Record field definitions:

Field Offset	Definition
PRO_B_RECD	Record type
PRO_W_PIX	Process index
PRO_W_NODE	Node number
PRO_F_CPUTIM	CPU time (10 Msec units)
PRO_F_PAGEFLTS	Soft page faults/CPUsec
PRO_F_PGFLTIO	Hard page fault IO/CPUsec
PRO_F_DIOS	Direct IO/sec
PRO_F_BIOS	Buffered IO/sec
PRO_F_GPGCNT	Global pages in WS count
PRO_F_PPGCNT	Process pages in WS count
PRO_F_WSSIZE	Working set size limit
PRO_F_DFWSCNT	Working set default
PRO_F_WSQUOTA	Working set quota
PRO_F_WSEXTENT	Working set extent
PRO_F_UPTIME	Runtime (secs)
PRO_F_IMGACTS	Image activations/sec
PRO_F_COMPU	Percentage of Time in COM state
PRO_W_RSN	Resource wait state mask
PRO_V_RSN_ASTWAIT	
PRO_V_RSN_MAILBOX	

Field Offset	Definition
PRO_V_RSN_NPDYNMEM	
PRO_V_RSN_PGFILE	
PRO_V_RSN_PGDYNMEM	
PRO_V_RSN_BRKTHRU	
PRO_V_RSN_IACLOCK	
PRO_V_RSN_JQUOTA	
PRO_V_RSN_LOCKID	
PRO_V_RSN_SWPFILE	
PRO_V_RSN_MPLEMPTY	
PRO_V_RSN_MPWBUSY	
PRO_V_RSN_SCS	
PRO_V_RSN_CLU	
PRO_W_SSS	Scheduler state mask
PRO_B_STATUS	Status (interactive=0/batch=1/network=2)
PRO_B_PRIB	Base priority
PRO_B_STATE	State (eg HIB, b0-b15)
PRO_B_PRI	Priority
PRO_B_IMGACT	Image activation(0=no/1=yes)
PRO_B_IMTRM	Image termination (0=no/1=yes)
PRO_B_LOGIN	Login (0=no/1=yes)
PRO_B_LOGOUT	Logout (0=no/1=yes)
PRO_B_KAST	Measurement characteristics
PRO_M_KAST PRO_V_KAST	Measurement KAST generated data
PRO_V_TT_ENDS PRO_M_TT_ENDS	Think time continues into next interval. These fields are stored in the PRO_B_KAST data cell.
PRO_V_RT_ENDS PRO_M_RT_ENDS	Response time continues into next interval. These fields are stored in the PRO_B_KAST data cell.
PRO_B_AWSA	Automatic Working Set Adjustment status (0=on/1=off)

Field Offset	Definition
PRO_B_PAGIDX	V4 paging index
PRO_B_SWPIDX	V4 swapping file index
PRO_A_IMAGENAME	Image name ASCID address
PRO_A_IMAGEDIR	Not used
PRO_A_USERNAME	Username ASCID address
PRO_L_PID	Extended process index
PRO_L_MWAIT	MWAIT hung reason code (NOT mask)
PRO_M_NOAST	Waiting for AST delivery
PRO_M_JIB	JIB quota exceeded
PRO_M_PCB	Waiting for PCB
PRO_M_IO	Waiting for I/O
PRO_M_IRP	Waiting for I/O request processing
PRO_M_DLOCK	Deadlock embrace
PRO_M_ENQ	Waiting for LOCK manager
PRO_M_UNK	Unknown reason
PRO_F_THRUPUT	Throughput/sec
PRO_F_OPS	Operations/sec
PRO_F_TAPE_IO	Tape I/O operations/sec
PRO_F_TAPE_THRUPUT	Tape throughput/sec
PRO_F_TERM_INPUT	Terminal input commands/sec
PRO_F_TERM_THRUPUT	Terminal throughput/sec
PRO_F_THINK_TIME	Think time. The time from the start of a terminal input to the completion of that terminal input or to the end of the interval or image termination. Unit of time is milliseconds.
PRO_F_RESPONSE_TIME	Response time. The time from the completion of a terminal input to the next terminal input or output, or to the end of the interval or image termination. Unit of time is milliseconds.
PRO_L_UIC	The processors UIC

Field Offset	Definition
PRO_A_DEVICE_1	Name of the top I/O disk or tape for a given process
PRO_F_OPS_1	Number of operations per second for the process on the disk or tape identified by device 1
PRO_A_DEVICE_2	Name of the second top I/O disk or tape for a given process
PRO_A_OPS_2	Number of operations per second for the process on the disk or tape identified by device 2
PRO_A_ACCOUNT	The 0-8 character UAF account name for the process.
PRO_A_PROCESS	The 0-15 character process name for the process.
PRO_F_VA_USED	The peak Virtual Address space used by the process.
PRO_F_COMMAND_WAIT	Either the duration of think time that continues into the next interval (if PRO_M_TT_ENDS is set) or the duration of the most recent completed think time within the interval (if PRO_M_TT_ENDS is clear). Unit of time is milliseconds.
PRO_F_RESPONSE_WAIT	Either the duration of response time that continues into the next interval (if PRO_M_RT_ENDS is set) or the duration of the most recent completed response time within the interval (if PRO_M_RT_ENDS is clear). Unit of time is milliseconds.
PRO_F_RESPONSE_TIME2	Response time in milliseconds. The time from the completion of a terminal input to the next terminal input, or to the end of the interval or image termination. This response time plus think time equals 100% of the interval.
PRO_L_PROCTYPE	Process type
PRO_V_INTER	Interactive process
PRO_V_BATCH	Batch process

Field Offset	Definition
PRO_V_NETWORK	Network process
PRO_V_DETACHED	Detached
PRO_V_SUBPROC	Subprocess
PRO_L_MPID	Master PID
PRO_A_IMAGEPATH	The device and directory specification where the image was activated

Metrics Record

The following table lists the Metrics Record Field Definitions:

Field Offsets	Definition
MET_B_TYPE	Record type (3)
MET_F_PROCCNT	Total number of processes
MET_F_COLPG	Average collided page state count
MET_F_MWAIT	Average MWAIT state count
MET_F_CEF	Average cluster event flag wait count
MET_F_PFW	Average page fault wait count
MET_F_LEF	Average local event flag wait count
MET_F_LEFO	Average outswapped local event flag wait count
MET_F_HIB	Average hibernate count
MET_F_HIBO	Average outswapped hibernate count
MET_F_SUSP	Average suspend count
MET_F_SUSPO	Average outswapped suspend count
MET_F_FPG	Average free page wait count
MET_F_COM	Average compute state count
MET_F_COMO	Average outswapped compute count
MET_F_CUR	Average current state count
MET_F_INTSTK	Percentage of Time on Interrupt stack
MET_F_KERNEL	Percentage of Time on Kernel stacks
MET_F_EXEC	Percentage of Time on Exec stacks

Field Offsets	Definition
MET_F_SUPER	Percentage of Time on Supervisor stacks
MET_F_USER	Percentage of Time on User stacks
MET_F_COMPAT	Percentage of Time in Compatibility mode
MET_F_IDLE	Percentage of Time Idle
MET_F_MP_SYNCH	Percentage of Time in MP synchronization
MET_F_INTSTK2	Time on Interrupt stack (not used)
MET_F_KERNEL2	Time on Kernel stacks 2 (not used)
MET_F_EXEC2	Time on Exec stacks 2 (not used)
MET_F_SUPER2	Time on Supervisor stack 2 (not used)
MET_F_USER2	Time on User stacks 2 (not used)
MET_F_COMPAT2	Time in Compatibility mode 2 (not used)
MET_F_IDLE2	Time Idle 2 (not used)
MET_F_FAULTS	Total page faults/sec
MET_F_PREADS	Page fault read/sec
MET_F_PREADIO	Page fault read I/O/sec
MET_F_PWRITES	Page fault write/sec
MET_F_PWRITIO	Page fault write I/O/sec
MET_F_FREFLTS	Page faults from free list/sec
MET_F_MFYFLTS	Page faults from modified list/sec
MET_F_DZROFLTS	Demand Zero page fault/sec
MET_F_GVALID	Global page fault/sec
MET_F_WRTINPROG	Transition page fault/sec
MET_F_SYSFAULTS	System page fault/sec
MET_F_FREECNT	Free list page count
MET_F_MFYCNT	Modified list page count
MET_F_DIRIO	Total Direct I/O/sec
MET_F_BUFIO	Total Buffered IO/sec
MET_F_MBREADS	Mailbox read/sec
MET_F_MBWRITES	Mailbox write/sec
MET_F_LOGNAM	Logical name translation/sec

Field Offsets	Definition
MET_F_ISWPCNT	Inswap/sec
MET_F_FCPTURN	File window miss/sec
MET_F_SPLIT	Split Transfers/sec
MET_F_HIT	Transfers w/o window turns/sec
MET_F_DIRHIT	Directory hits/sec
MET_F_DIRMISS	Directory misses/sec
MET_F_QUOHIT	Quota cache hits/sec
MET_F_QUOMISS	Quota cache misses/sec
MET_F_FIDHIT	File ID cache hits/sec
MET_F_FIDMISS	File ID cache misses/sec
MET_F_EXTHIT	Extent cache hits/sec
MET_F_EXTMISS	Extent cache misses/sec
MET_F_FILHDR_HIT	File header cache hits/sec
MET_F_FILHDR_MISS	File header cache misses/sec
MET_F_DIRDATA_HIT	Directory data block hits/sec
MET_F_DIRDATA_MISS	Directory data block misses/sec
MET_F_STORAGMAP_HIT	Storage bit map cache hits/sec
MET_F_STORAGMAP_MISS	Storage bit map cache misses/sec
MET_F_OPENS	Files open/sec
MET_F_ERASEIO	Erase QIOs/sec
MET_F_VOLLCK	XQP volume synchronization locks/sec
MET_F_VOLWAIT	Times XQP had to wait for volume synchronization lock/sec
MET_F_SYNCHLCK	XQP Directory and volume synchronization locks/sec
MET_F_SYNCHWAIT	Times XQP had to wait for a directory and volume synchronization lock/sec
MET_F_ACCLCK	XQP access locks/sec
MET_F_XQPCACHEWAIT	Times XQP had to wait for cache free space /sec
MET_F_FILECPU	Percentage CPU time in file system
MET_F_ARRLOCPK	DECNET arriving local packets/sec

Field Offsets	Definition
MET_F_DEPLOCPK	DECNET departing local packets/sec
MET_F_ARRTRAPK	DECNET transit packets/sec
MET_F_TRCNGLOS	DECNET transit congestion loss/sec
MET_F_RCVBUFFL	DECNET receiver buffer failures/sec
MET_F_ENQNEW_LOC	Local new requests/sec
MET_F_ENQNEW_IN	Incoming new requests/sec
MET_F_ENQNEW_OUT	Outgoing new requests/sec
MET_F_ENQCVT_LOC	Local conversion requests/sec
MET_F_ENQCVT_IN	Incoming conversion requests/sec
MET_F_ENQCVT_OUT	Outgoing conversion requests/sec
MET_F_DEQ_LOC	Local dequeue requests/sec
MET_F_DEQ_IN	Incoming dequeue requests/sec
MET_F_DEQ_OUT	Outgoing dequeue requests/sec
MET_F_ENQWAIT	Enqueue requests waiting/sec
MET_F_ENQNOTQD	Enqueue requests not queued/sec
MET_F_BLK_LOC	Local blocking ASTs queued/sec
MET_F_BLK_IN	Incoming blocking ASTs queued/sec
MET_F_BLK_OUT	Outgoing blocking ASTs queued/sec
MET_F_DIR_IN	Incoming directory operations/sec
MET_F_DIR_OUT	Outgoing directory operations/sec
MET_F_DLCKMSGS_IN	Incoming deadlock detection messages/sec
MET_F_DLCKMSGS_OUT	Outgoing deadlock detection messages/sec
MET_F_DLCKSRCH	Deadlock searches/sec
MET_F_DLCKFND	Deadlock finds/sec
MET_F_USERPAGES	User memory pages
MET_F_TREADS	Terminal reads (unused)
MET_F_TWRITES	Terminal writes (unused)
MET_F_IMGACTS	Image activation/sec
MET_F_IMGTRMS	Image termination/sec

Field Offsets	Definition
MET_F_LOCK_MAX	Lock ID table length
MET_F_LOCK_CNT	Lock IDs in use
MET_F_RESOURCE_MAX	Resource table length
MET_F_RESOURCE_CNT	Resources in use
MET_F_NP_POOL_MAX	Non-paged pool length
MET_F_NP_FREE_BLOCKS	Non-paged free blocks
MET_F_NP_FREE_LEQU_32	Non-paged free leq 32 bytes
MET_F_NP_FREE	Non-paged free bytes
MET_F_NP_MAX_BLOCK	Non-paged maximum block
MET_F_NP_MIN_BLOCK	Non-paged minimum block
MET_F_PG_POOL_MAX	Paged pool length
MET_F_PG_FREE_BLOCKS	Paged free blocks
MET_F_PG_FREE_LEQU_32	Paged free leq 32 bytes
MET_F_PG_FREE	Paged free bytes
MET_F_PG_MAX_BLOCK	Paged maximum block
MET_F_PG_MIN_BLOCK	Paged minimum block
MET_F_SRP_MAX	SRP list length *
MET_F_SRP_CNT	SRPs in use *
MET_F_IRP_MAX	IRP list length *
MET_F_IRP_CNT	IRPs in use *
MET_F_LRP_MAX	LRP list length *
MET_F_LRP_CNT	LRPs in use *
MET_F_RDT_MAX	I/O request description table size
MET_F_RDT_QUE	I/O request description table queue
MET_F_MSCP_BUFF_MAX	MSCP number original buffers
MET_F_MSCP_BUFF_FREE	MSCP number free buffers
MET_F_MSCP_BUFF_MIN	MSCP smallest buffer allowed
MET_F_MSCP_BUFF_AVL	MSCP number free pool bytes
MET_F_MSCP_PACK_MAX	MSCP number original packets
MET_F_MSCP_PACK_FREE	MSCP number free packets

Field Offsets	Definition
MET_F_MSCP_BUFF_QUE	MSCP buffer wait queue
MET_F_MSCP_BUFF_PEAK	MSCP wait queue highwater mark
MET_F_MSCP_IO_SPLITS	MSCP number split transfers/sec
MET_F_MSCP_IO_FRAGMENTS	MSCP number IO fragments/sec
MET_F_MSCP_OPCOUNT	MSCP operation/sec
MET_F_MSCP_READ_CNT	MSCP read/sec
MET_F_MSCP_WRITE_CNT	MSCP write/sec
MET_F_PAGING_TOTAL	Paging total pages
MET_F_PAGING_FREE	Paging free pages
MET_F_OSWPCNT	Out swaps
MET_F_HISWPCNT	Header in swaps
MET_F_HOSWPCNT	Header out swaps
MET_F_BADPAGE_FAULTS	Bad page faults
MET_F_TRANSFLTS	Transition faults
MET_F_OPEN_FILES	Number of files open
MET_F_INTERACTIVE	Interactive processes
MET_F_NETWORK	Network processes
MET_F_BATCH	Batch processes
MET_F_SPMSAMPCNT	Sample count
MET_F_SPMBUSY	Busy
MET_F_SPMSWPBUSY	Swap busy
MET_F_SPMIOBUSY	MIO busy
MET_F_SPMANYIOBUSY	Any IO busy
MET_F_SPMPAGEWAIT	Page wait
MET_F_SPMSWAPWAIT	Swap wait
MET_F_SPMMMGWAIT	MMG wait
MET_F_SPMSYSIDLE	SYS idle
MET_F_SPMCPUONLY	CPU only
MET_F_SPMIOONLY	IO only
MET_F_SPMCPUIO	CPU IO

Field Offsets	Definition
MET_F_SPMAVAILCPU	Available CPUs sample space count
MET_F_VBSSCPUTICK	CPU time used by VBS transitions
MET_F_CACHE_SIZE	The Virtual I/O cache size in pages.
MET_F_CACHE_FREE	The number of Virtual I/O cache free pages.
MET_F_CACHE_USED	The number of Virtual I/O cache pages in use.
MET_F_CACHE_MAXI	The Virtual I/O cache maximum size in pages.
MET_F_CACHE_FILES	The number of Virtual I/O cache files retained.
MET_F_CACHE_RDIO	The Virtual I/O cache read I/O rate, reads/second
MET_F_CACHE_READHITS	The Virtual I/O cache read hit rate, read hits/second
MET_F_CACHE_WRIO	The Virtual I/O cache write I/O rate, writes/second
MET_F_CACHE_BYPASS	The Virtual I/O cache bypassing rate, I/Os bypassing the cache/second
MET_F_CACHE_WRITEHITS	The Virtual I/O cache write hit rate, write hits/second
MET_F_CACHE_MISS_LT33	The rate of reads that bypass the I/O cache, less than 33 blocks each
MET_F_CACHE_MISS_3364	The rate of reads that bypass the I/O cache, 33 blocks through 64 blocks each
MET_F_CACHE_MISS_65127	The rate of reads that bypass the I/O cache, 65 blocks through 127 blocks each
MET_F_CACHE_MISS_128255	The rate of reads that bypass the I/O cache, 128 blocks through 255 blocks each
MET_F_CACHE_MISS_GT255	The rate of reads that bypass the I/O cache, greater than 255 blocks each

* For OpenVMS Version 6.0 these fields are obsolete and are set to zero.

CPU Record

The following table lists the CPU Record field definitions:

Field Offsets	Definition
CPU_B_TYPE	Record type (10)
CPU_L_PHY_CPUID	Physical ID
CPU_L_STATE	Active state code
CPU_C_RUN	Run state
CPU_L_STATUS	Status code
CPU_C_PRIMID	Primary CPU
CPU_F_KERNEL	Percentage of Kernel time
CPU_F_EXEC	Percentage of Executive time
CPU_F_SUPER	Percentage of Supervisor time
CPU_F_USER	Percentage of User time
CPU_F_INTERRUPT	Percentage of Interrupt time
CPU_F_COMPAT	Percentage of Compatibility time
CPU_F_NULL	Percentage of NULL time
CPU_F_MP_SYNCH	Percentage of SMP synchronization in Kernel mode

Parameter Record

The following table lists the Parameter Record field definitions:

Field Offsets	Definition
PAR_B_TYPE	Record type (2)
PAR_F_DORMANTWAIT	SYSGEN parameter DORMANTWAIT
PAR_F_GBLPAGES	SYSGEN parameter GBLPAGES
PAR_F_DEFPRI	SYSGEN parameter DEFPRI
PAR_F_MAXPROCESSCNT	SYSGEN parameter MAXPROCESSCNT
PAR_F_SPTREQ	SYSGEN parameter SPTREQ
PAR_F_LRPCOUNT	SYSGEN parameter LRPCOUNT

Field Offsets	Definition
PAR_F_LRPCOUNTV	SYSGEN parameter LRPCOUNTV
PAR_F_LRPSIZE	SYSGEN parameter LRPSIZE
PAR_F_SRPCOUNT	SYSGEN parameter SRPCOUNT
PAR_F_LOCKDIRWT	SYSGEN parameter LOCKDIRWT
PAR_F_MINWSCNT	SYSGEN parameter MINWSCNT
PAR_F_SYSMWCNT	SYSGEN parameter SYSMWCNT
PAR_F_BALSETCNT	SYSGEN parameter BALSETCNT
PAR_F_IRPCOUNT	SYSGEN parameter IRPCOUNT
PAR_F_IRPCOUNTV	SYSGEN parameter IRPCOUNTV
PAR_F_WSMAX	SYSGEN parameter WSMAX
PAR_F_NPAGEDYN	SYSGEN parameter NPAGEDYN
PAR_F_NPAGEVIR	SYSGEN parameter NPAGEVIR
PAR_F_PAGEDYN	SYSGEN parameter PAGEDYN
PAR_F_SRPCOUNTV	SYSGEN parameter SRPCOUNTV
PAR_F_SRPSIZE	SYSGEN parameter SRPSIZE
PAR_F_QUANTUM	SYSGEN parameter QUANTUM
PAR_F_MPW_HILIMIT	SYSGEN parameter MPW_HILIMIT
PAR_F_MPW_LOLIMIT	SYSGEN parameter MPW_LOLIMIT
PAR_F_MPW_THRESH	SYSGEN parameter MPW_THRESH
PAR_F_MPW_WAITLIMIT	SYSGEN parameter MPW_WAITLIMIT
PAR_F_PFRATL	SYSGEN parameter PFRATL
PAR_F_PFRATH	SYSGEN parameter PFRATH
PAR_F_CTLFLAGS	SYSGEN parameter MMG_CTLFLAGS
PAR_F_WSINC	SYSGEN parameter WSINC
PAR_F_WSDEC	SYSGEN parameter WSDEC
PAR_F_AWSMIN	SYSGEN parameter AWSMIN
PAR_F_AWSTIME	SYSGEN parameter AWSTIME
PAR_F_SWPRATE	SYSGEN parameter SWPRATE
PAR_F_SWPOUTPGCNT	SYSGEN parameter SWPOUTPGCNT
PAR_F_SWPALLOCINC	SYSGEN parameter SWPALLOCINC

Field Offsets	Definition
PAR_F_LONGWAIT	SYSGEN parameter LONGWAIT
PAR_F_FREELIM	SYSGEN parameter FREELIM
PAR_F_FREEGOAL	SYSGEN parameter FREEGOAL
PAR_F_GROWLIM	SYSGEN parameter GROWLIM
PAR_F_BORROWLIM	SYSGEN parameter BORROWLIM
PAR_F_ACP_MAPCACHE	SYSGEN parameter ACP_MAPCACHE
PAR_F_ACP_HDRCACHE	SYSGEN parameter ACP_HDRCACHE
PAR_F_ACP_DIRCACHE	SYSGEN parameter ACP_DIRCACHE
PAR_F_ACP_WORKSET	SYSGEN parameter ACP_WORKSET
PAR_F_ACP_DINDXCACHE	SYSGEN parameter ACP_DINDXCACHE
PAR_F_ACP_FIDCACHE	SYSGEN parameter ACP_FIDCACHE
PAR_F_ACP_EXTCACHE	SYSGEN parameter ACP_EXTCACHE
PAR_F_ACP_QUOCACHE	SYSGEN parameter ACP_QUOCACHE
PAR_F_ACP_EXTLIMIT	SYSGEN parameter ACP_EXTLIMIT
PAR_F_PFCDEFAULT	SYSGEN parameter PFCDEFAULT
PAR_F_MPW_WRTCLUSTER	SYSGEN parameter MPW_WRTCLUSTER
PAR_F_IOTA	SYSGEN parameter IOTA
PAR_F_PIXSCAN	SYSGEN parameter PIXSCAN
PAR_F_PHYSICALPAGES	SYSGEN parameter PHYSICALPAGES
PAR_F_LOCKIDTBL	SYSGEN parameter LOCKIDTBL
PAR_F_RESHASHTBL	SYSGEN parameter RESHASHTBL
PAR_F_GBLSECTIONS	SYSGEN parameter GBLSECTIONS
PAR_F_DEADLOCK_WAIT	SYSGEN parameter DEADLOCK_WAIT
PAR_F_MSCP_CREDITS	SYSGEN parameter MSCP_CREDITS
PAR_F_MPW_LOWAITLIMIT	SYSGEN parameter MPW_LOWAITLIMIT
PAR_F_MPW_IOLIMIT	SYSGEN parameter MPW_IOLIMIT
PAR_F_SCSBUFFCNT	SYSGEN parameter SCSBUFFCNT
PAR_F_SCSCONNCNT	SYSGEN parameter SCSCONNCNT

Field Offsets	Definition
PAR_F_SCSRESPCNT	SYSGEN parameter SCSRESPCNT
PAR_F_SCSMAXDG	SYSGEN parameter SCSMAXDG
PAR_F_SCSMAXMSG	SYSGEN parameter SCSMAXMSG
PAR_F_POOLCHECK	SYSGEN parameter POOLCHECK
PAR_F_MULTIPROC	SYSGEN parameter MULTIPROCESSING
PAR_F_VBSSENA	SYSGEN parameter VBSS_ENABLE
PAR_F_CACHE_STATE	The state of the Virtual I/O cache. The cache may be in one of the following states that preclude the collection of cache statistics:
PAR_M_CACHE_UNKVER = 1	The version of the cache was not recognized by the data collector so no cache statistics could be collected.
PAR_M_CACHE_HETERO = 2	The cluster had a mixed configuration of nodes one or more of which did not have caching capability so no cache statistics were available to be collected.
PAR_M_CACHE_DISABLED = 4	The cache was not enabled so no cache statistics were available to be collected.
PAR_F_LCKMGR_MODE	SYSGEN parameter LCKMGR_MODE
PAR_F_SMP_CPUS	SYSGEN parameter SMP_CPUS
PAR_F_LOAD_SYS_IMAGES	SYSGEN parameter LOAD_SYS_IMAGES
PAR_F_PQL_DWSDEFAULT	SYSGEN parameter DWSDEFAULT
PAR_F_PQL_MWSDEFAULT	SYSGEN parameter MWSDEFAULT
PAR_F_PQL_DWSQUOTA	SYSGEN parameter DWSDEFAULT
PAR_F_PQL_MWSQUOTA	SYSGEN parameter MWSDEFAULT
PAR_F_PQL_DWSEXTENT	SYSGEN parameter DWSEXTENT
PAR_F_PQL_MWSEXTENT	SYSTEM parameter MWSEXTENT

Time Record

The following table lists the Time Record field definitions:

Field Offsets	Definition
TIM_B_TYPE	Record type (1)
TIM_B_NODE_NAME	Node name length
TIM_T_NODE_NAME	Node name string (14 character max)
TIM_B_NODE_NUMBER	Node number
TIM_L_SID	System ID number
TIM_L_HWTYPE	Node hardware type
TIM_W_INTERVAL	Sampling interval
TIM_W_INTEGRATION	Integration interval
TIM_Q_START_TIME	Start of interval system time
TIM_Q_END_TIME	End of interval system time
TIM_F_UPTIME	Uptime during interval (seconds)
TIM_Q_SWVERS	OpenVMS software version
TIM_B_HWNAME_LEN	Hardware name length
TIM_T_HWNAME	Hardware name (24 char)
TIM_W_HWMODEL	Hardware model number
TIM_W_MAX_PROCESSES	Maximum processes possible

Device Record

The following table lists the Device Record field definitions:

Field Offsets	Definition
DEV_B_TYPE	Record type (5)
DEV_B_DEVCLASS	Device class (disk)
DEV_A_VOLNAME	Volume name ASCID address
DEV_A_NODENAME	Node name ASCID address
DEV_B_NODE_NUMBER	Node number
DEV_B_CIO_REQUESTOR	CIO requestor

Field Offsets	Definition
DEV_A_HWTYPE	Hardware type (HS50,V780,...)
DEV_L_ALLOCLS	Allocation class
DEV_A_CTLR_NAME	Controller ASCID address
DEV_W_UNIT	Unit number
DEV_B_DEV_TYPE	Type of device (see $DCDEF)
DEV_B_ADP_TYPE	Adapter type (MBA or UBA)
DEV_B_ADP_NUMBER	Adapter number
DEV_W_ADP_TR	Adapter Nexus number
DEV_B_PDT_TYPE	Port type (UDA,CI,NI,Passthru)
DEV_B_DEVCHAR	Device characteristics
DEV_M_DUA	Dual ported
DEV_M_MNT	Mounted
DEV_M_CLU	Available clusterwide
DEV_M_CDP	Dual pathed with 2 UCBs
DEV_M__2P	Two known paths
DEV_M_MSCP	Accessed using MSCP
DEV_M_SRV	Served by MSCP server
DEV_F_SERVICE	Average milliseconds response time / operation
DEV_F_QLEN	Queue length
DEV_F_OPCNT	Operations/sec
DEV_F_IOCNT	Throughput byte/sec
DEV_F_PAGOP	Paging operations/sec
DEV_F_PAGIO	Paging throughput/sec
DEV_F_SWPOP	Swapping operations/sec
DEV_F_SWPIO	Swapping throughput/sec
DEV_F_BUSY	Disk busy time
DEV_F_ITVL	Measurement interval (milliseconds)
DEV_F_ERRCNT	Error count
DEV_B_PAGIDX	V4 paging file index
DEV_B_SWAPIDX	V4 swapping file index

Field Offsets	Definition
DEV_F_RDCNT	Read count/sec
DEV_F_FREE	Free block count
DEV_B_SET	Shadow status
DEV_M_COPYING	Shadow copying (1b0=yes) *
DEV_V_MBR	Disk is either a shadow set, stripe set, or bound volume set member
DEV_V_MAS	Disk is either a shadow set, stripe set, or bound volume set master.
DEV_V_HSC	Disk is a member or master of an HSC-based shadow set.
DEV_V_HBS	Disk is a member or master of a host-based shadow set.
DEV_V_VOL	Disk is a member or master of a bound volume set.
DEV_V_STR	Disk is either a member or master of a stripe set.
DEV_F_DINDXSIZE	Directory index cache pages
DEV_F_QUOSIZE	Quota cache entries
DEV_F_FIDSIZE	FID cache entries
DEV_F_EXTSIZE	Extent cache entries
DEV_F_HDRSIZE	File header cache pages
DEV_F_DIRSIZE	Directory data cache pages
DEV_F_MAPSIZE	Storage bitmap cache pages
DEV_A_CACHENAME	cache name ASCID address
DEV_F_MSCPOP	MSCP operations count /sec
DEV_F_MSCPPG	MSCP paging/swapping count /sec
DEV_A_HWNAME	V5.0 node hardware ASCID address
DEV_A_DEVNAME	Device name ASCID address
DEV_L_ROOT	Set membership identifier **
DEV_F_MSCPIO	MSCP throughput byte/sec
DEV_F_SPLIT	Split I/O count
DEV_W_REQ	K.SDI requestor number
DEV_W_PORT	K.SDI port number

Field Offsets	Definition
DEV_F_MAXBLOCK	Maximum number of blocks
DEV_A_DEVNAME	Device names

* If both DEV_V_MBR and DEV_V_MAS set, then DEV_V_MAS pertains to DEV_V_VOL, DEV_V_STR, DEV_V_HBS, or DEV_V_HSC, in that order, whichever two are set.

** Common bytes 2-4 indicate membership in same volume, stripe, or shadow set. High order of byte 1 indicates membership in same stripe set if DEV_V_STR is set. Low order of byte 1 indicates membership in same shadowset if DEV_V_HSC or DEV_V_HBS is set.

Tape Record

The following table lists the Tape Record field definitions:

Field Offsets	Definition
MAG_B_TYPE	Record type (9)
MAG_B_DEVCLASS	Device class
MAG_A_VOLNAME	Volume name string desc. addr
MAG_A_NODENAME	Node name ASCID addr
MAG_B_NODE_NUMBER	Node number
MAG_A_HWTYPE	Hardware type
MAG_L_ALLOCLS	Allocation class
MAG_A_CTLR_NAME	Controller ASCID addr
MAG_W_UNIT	Unit number
MAG_B_DEV_TYPE	Type of device (see $DCDEF)
MAG_B_ADP_TYPE	Adapter type (MBA or UBA)
MAG_B_ADP_NUMBER	Adapter number
MAG_W_ADP_TR	Adapter Nexus number
MAG_B_PDT_TYPE	Port type
MAG_B_DEVCHAR	Characteristics BIT vector (NOT MASK)
MAG_F_OPCNT	Operations /sec
MAG_F_BUSY	Busy time

Field Offsets	Definition
MAG_F_ITVL	Measurement interval (ms)
MAG_F_ERRCNT	Error count
MAG_A_DEVNAME	Device name ASCID address

Communications Record

The following table lists the Communications Record field definitions:

Field Offsets	Definition
COM_B_TYPE	Record type
COM_B_DEVCLASS	Device class
COM_A_CTLR_NAME	Controller ASCID addr
COM_W_UNIT	Unit number
COM_B_DEV_TYPE	Type of device
COM_B_DEVCHAR	Characteristics
COM_F_OPCNT	Operations /sec
COM_A_DEVNAME	Device name ASCID address

Hot File

The following table lists the Hot File Record field definitions:

Field Offsets	Definition
FIL_B_TYPE	Record type (10)
FIL_A_DEVICE	Disk device name
FIL_A_DIRECTORY	Directory name
FIL_A_FILE	File name *
FIL_W_FID_NUM	FID file number
FIL_W_FID_SEQ	FID sequence number
FIL_W_FID_RVN	FID relative volume number
FIL_W_INDEX	Disk index number

Field Offsets	Definition
FIL_F_SERVICE	Service time (milliseconds)
FIL_F_OPCNT	Operations count (incremental)
FIL_F_IOCNT	Throughput byte count (incremental)
FIL_F_RDCNT	Read operations
FIL_F_SPLITS	Split I/Os
FIL_F_TURNS	Window turns (unused)
FIL_F_PAGOP	Paging operations
FIL_F_SWPOP	Swapping operations
FIL_F_MSCPOP	MSCP operations
FIL_L_PID	Process index

* Use of CTX$V_FIDDLE causes the FID or Non-Virtual Queue I/O to be specified if the name FIL_A_FILE, in a hot file record, is blank (see PSPA$OPEN_CTX).

Appendix A: Performance Manager Sample Programs

Included with the Performance Manager are sample programs built to extract data from the daily data files. You can use the sample programs as examples or as templates to build upon. The sample programs are located in the PSPA$EXAMPLES directory, which was automatically created when Performance Manager was installed, and are available in the following languages:

- HP C—PSPA$GETDATA.C
- HP Pascal—PSPA$GETDATA.PAS
- OpenVMS MACRO—PSPA$GETDATA.MAR

The PSPA$GETDATA.COM file has the necessary commands to compile and link the sample programs and a sample script to execute the programs.

Index

A

adapter types • 75, 78

C

callable routines
- PSPA$OPEN_CONTEXT • 12, 38, 53
- PSPA$READ_CONTEXT • 37

condition handling • 7

context block
- definition • 7
- offsets to fields • 58
- total length defined • 9
- usage, Bliss example • 10

D

DECnet Phase V support, CIFDE node name translation • 13

default
- PSPA$READ.EXE • 57
- working set • 60

H

hard page fault, process record field • 60

I

interval
- integration • 75
- measurement • 75, 78
- record, PSPA$READ_CONTEXT • 37
- sampling • 75

L

library file for CIFDE data structure definitions • 57

M

MSCP
- device record fields • 75
- metric record fields • 64

P

phase V support, CIFDE node name translation • 13

procedure calling • 7

process record field • 60

programs, user-written • 7

PSPA$GETDATA.COM, sample macro program • 81

PSPA$LIB, CIFDE library files • 57

PSPA$READ.EXE executes CIFDE procedures • 57

Q

queue length, device record field • 75

S

soft page fault, process record field • 60