AA-PB90A-TE

DECtrace for VMS

User's Guide

June 1990

This manual describes the DECtrace for VMS software and how to use it to collect and report on event-based data.

Operating System:	VMS
Software Version:	DECtrace for VMS Version 1.0

digital equipment corporation
maynard, massachusetts

First Printing, June 1990

This document was prepared using VAX DOCUMENT, Version 1.2.

Contents

Preface xiii

1 Introduction to DECtrace

1.1 What Is Event-Based Data Collection? 1–1
1.2 What Can DECtrace Do for You? 1–1
1.3 How Does DECtrace Work? 1–2
1.3.1 DECtrace Registrar Process 1–3
1.3.2 DECtrace Administration Database 1–3
1.4 Overview of DECtrace Processing 1–4
1.5 Explanation of Sample Application 1–5

2 Creating a Facility Selection

2.1 Choosing Which Data to Collect 2–2
2.2 Creating a Selection 2–3
2.3 Deleting a Selection 2–4
2.4 Displaying Information About a Selection 2–5
2.4.1 BRIEF Format 2–5
2.4.2 FULL Format 2–6
2.4.3 NAMES_ONLY Format 2–7

3 Scheduling Data Collection

3.1 Choosing the Processes from Which to Collect Data 3–2
3.1.1 Process Registration 3–2
3.1.2 Using the Registration ID 3–3
3.1.3 Displaying Information about Process Registration 3–5
3.2 Scheduling a Collection 3–8
3.2.1 Scheduling Data Collection on a Standalone System 3–9
3.2.2 Scheduling Data Collection on a Cluster 3–9
3.2.3 Scheduling Data Collection on Part of a Cluster 3–9
3.3 Data Collection Files 3–11
3.3.1 File Protection Schemes 3–12
3.4 Canceling Data Collection 3–13
3.5 Displaying Information about Data Collection 3–14
3.5.1 Displaying the Schedule for Data Collection 3–14
3.5.2 Displaying the History for Collections 3–15

4 Generating Reports

4.1 Overview of Reporting 4–2
4.2 Merging and Formatting Your Data Files 4–2
4.2.1 Merging and Formatting the Collected Data into an Rdb/VMS Database 4–2
4.2.2 Merging and Formatting the Collected Data into a VAX RMS File 4–3
4.3 Formatting Optimization 4–3
4.4 Report Types 4–4
4.4.1 Detail Report 4–5
4.4.2 Frequency Report 4–9
4.4.3 Summary Report 4–11
4.5 Text Wrapping in Reports 4–18
4.5.1 Groupable Items 4–18
4.5.2 Groups 4–18
4.5.3 Headers 4–19
4.5.4 Report Item Text 4–20
4.6 Special Cases for Reporting on Duration Events 4–20
4.7 Report Optimization 4–22
4.8 Report Options 4–22
REPORT OPTIONS—EVENT 4–24
REPORT OPTIONS—ITEM 4–27
REPORT OPTIONS—RESTRICTION 4–28

5 Instrumenting Applications

5.1 Programming Interface 5–2
5.2 DECtrace Data Structures 5–4
5.3 Determining Your Events 5–5
5.4 Using EPC$INIT to Register a Facility 5–6
5.4.1 Waiting for EPC$INIT to Complete 5–9
5.5 Coding an Event in the ATM Sample Application 5–10
5.6 Instrumenting an Application in VAX C 5–11
5.6.1 Instrumenting Simple Events 5–12
5.6.2 Instrumenting Events and Items 5–13
5.6.2.1 Collecting Basic Facility-Specific Items 5–14
5.6.2.2 Collecting Facility-Specific Items Efficiently 5–19
5.6.2.3 Collecting Facility-Specific Items with Maximum Efficiency 5–20
5.7 Instrumenting a Multi-Threaded Facility Using VAX BLISS-32 5–28
5.7.1 Call EPC$INIT to Register the NEW_TOOL Facility 5–28
5.7.2 Call EPC$SET_CONTEXT to Set Thread Context 5–30
5.7.3 Call EPC$DELETE_CONTEXT to Delete the Thread Context 5–31
5.7.4 Call EPC$START_EVENT and EPC$END_EVENT to Collect Event Data 5–31
5.8 Linking an Instrumented Program 5–32

6 Creating Facility Definitions

6.1 Creating a Facility Definition 6–2
6.1.1 Creating and Defining Events 6–3
6.1.2 Creating and Defining Items 6–5
6.1.3 Creating and Defining Collection Classes 6–8
6.2 Deleting Facility Definitions 6–9
6.3 Transporting Facility Definitions 6–10
6.3.1 Extracting Definitions 6–10
6.3.2 Inserting Definitions Using a KITINSTAL.COM Procedure 6–11
6.3.3 Inserting Definitions Without KITINSTAL.COM 6–12
6.4 Displaying Facility Definitions 6–13
6.4.1 FULL Format 6–14
6.4.2 NAMES_ONLY Format 6–19
6.5 Facility Definition Options 6–20
ITEM 6–22
GROUP 6–26
EVENT 6–27
CLASS 6–30
DEFAULT_CLASS 6–31

7 DECtrace Commands
@ (EXECUTE PROCEDURE) 7–4
CANCEL COLLECTION 7–5
CREATE DEFINITION 7–8
CREATE DEFINITION OPTIONS—ITEM 7–11
CREATE DEFINITION OPTIONS—GROUP 7–15
CREATE DEFINITION OPTIONS—EVENT 7–16
CREATE DEFINITION OPTIONS—CLASS 7–19
CREATE DEFINITION OPTIONS—DEFAULT_CLASS 7–20
CREATE SELECTION 7–21
DELETE DEFINITION 7–25
DELETE SELECTION 7–27
EXIT 7–29
EXTRACT DEFINITION 7–30
FORMAT 7–32
HELP 7–39
INSERT DEFINITION 7–40
REPORT 7–42
REPORT OPTIONS—EVENT 7–47
REPORT OPTIONS—ITEM 7–51
REPORT OPTIONS—RESTRICTION 7–52
SCHEDULE COLLECTION 7–54
SET HISTORY 7–59
SHOW COLLECTION 7–61
SHOW DEFINITION 7–63
SHOW HISTORY 7–65
SHOW REGISTER 7–68
SHOW SELECTION 7–69
SHOW VERSION 7–71
SPAWN 7–72
STOP SYSTEM 7–76

8 DECtrace Service Routines

8.1 Error Handling 8–2
EPC$DELETE_CONTEXT 8–4
EPC$END_EVENT 8–7
EPC$END_EVENTW 8–11
EPC$EVENT 8–12
EPC$EVENTW 8–16
EPC$INIT 8–17
EPC$SET_CONTEXT 8–23
EPC$START_EVENT 8–26
EPC$START_EVENTW 8–30

9 System Management Tasks

9.1 Required Account and Process Quotas 9–1
9.2 Controlling DECtrace 9–2
9.2.1 Starting DECtrace 9–2
9.2.2 Stopping DECtrace 9–3
9.2.3 Recovering From a System Crash 9–3
9.3 Installing Facilities on Your System 9–4
9.3.1 Installing New Facilities 9–4
9.3.2 Installing New Versions of Existing Facilities 9–4
9.3.3 Installing a New Version of Rdb/VMS 9–5
9.4 Managing the History Database 9–5
9.5 Disabling Data Collection 9–6
9.6 Improving DECtrace Performance 9–7
9.7 Troubleshooting DECtrace 9–7
9.7.1 Error Messages and Recovery Procedures 9–9

A Formatted Database and File Layouts

A.1 The Rdb/VMS Database A–1
A.1.1 The Data Types A–2
A.1.2 String Data Segmentation A–3
A.1.3 Relations for the Control Information A–4
A.1.3.1 The EPC$IDENT Relation A–5
A.1.3.2 The EPC$SELECTION Relation A–5
A.1.3.3 The EPC$COLLECTION Relation A–5
A.1.3.4 The EPC$DCF Relation A–6
A.1.3.5 The EPC$REG Relation A–7
A.1.3.6 The EPC$FACILITY Relation A–7

A.1.3.7 The EPC$EVENT Relation A–8
A.1.3.8 The EPC$ITEM Relation A–8
A.1.3.9 The EPC$EVENT_ITEM Record A–10
A.1.3.10 The EPC$PROCESS Relation A–11
A.1.3.11 The EPC$IMAGE Relation A–12
A.1.3.12 The EPC$DCF_IMAGE Relation A–13
A.1.3.13 The EPC$FACILITY_IMAGE Relation A–13
A.1.4 Relations for the Collected Data A–14
A.1.5 The Entity Relationship Diagram for the Database A–15
A.1.6 The Predefined Views A–15
A.1.6.1 Relating Collection Information with Event Data A–15
A.1.6.2 Relating Selection and Collection Information with Event Data A–15
A.1.6.3 Relating Image Information with Event Data A–16
A.1.6.4 Relating Process and Image Information with Event Data A–17
A.1.7 Limitations A–17
A.2 The VAX RMS File Format A–17
A.2.1 Data Types A–18
A.2.1.1 Date A–18
A.2.1.2 Fixed ASCIC/ASCIW String A–18
A.2.2 Records Organization A–18
A.2.2.1 Control Records A–19
A.2.2.1.1 FMTDICT Record A–20
A.2.2.1.2 COLLECTION Record A–20
A.2.2.1.3 FACILITY Record A–20
A.2.2.1.4 DCF Record A–21
A.2.2.1.5 IMAGE Record A–21
A.2.2.1.6 FACILITY-REGISTRATION Record A–23
A.2.2.1.7 EVENT Record A–24
A.2.2.1.8 Item Descriptor A–24
A.2.2.2 Data Collection Records A–26

Glossary

Index

Examples

2–1 Display Format for SHOW DEFINITION /FORMAT=NAMES_ONLY 2–3
2–2 Display for SHOW SELECTION Using the Brief Format 2–6
2–3 Display for SHOW SELECTION Using the Full Format 2–7
2–4 Display for SHOW SELECTION Using the Names Only Format 2–7
3–1 Process Registration Display 3–4
3–2 Display Format for SHOW REGISTER/CLUSTER 3–6
3–3 Display Format for SHOW REGISTER/NOCLUSTER 3–7
3–4 Sample Command Procedure for Starting a Local Collection 3–10
3–5 Display for SHOW COLLECTION Using the Brief Format 3–15
3–6 Display for SHOW HISTORY /FORMAT=ERROR 3–16
3–7 Display for SHOW HISTORY /FORMAT=INFORMATIONAL 3–17
4–1 Display for Detail Report 4–5
4–2 Sample Detail Report Based on ATM Data 4–6
4–3 Display for Frequency Report Using /INTERVAL=SECONDS 4–10
4–4 Sample Frequency Report Based on ATM Data 4–11
4–5 Display for Summary Report 4–14
4–6 Sample Summary Report Based on ATM Data 4–16
4–7 Wrapping of a Groupable Item 4–18
4–8 Wrapping of Groups 4–19
4–9 Wrapping of Headers 4–20
4–10 Wrapping of Report Item Text 4–20
5–1 Instrumentation of EPC$INIT in the ATM Application 5–8
5–2 Code to Guarantee Collection of All Events 5–9
5–3 Instrumentation of the WITHDRAWAL Event in the ATM Application 5–10
5–4 DB Sample Facility 5–15
5–5 DB Sample Facility—Performance Optimized 5–24
5–6 Instrumentation Using LIB$FIND_IMAGE_SYMBOL 5–33
6–1 Sample Procedure to Insert Binary Facility Definitions 6–11
6–2 Display for SHOW DEFINITION Using the Full Format 6–15
6–3 Display for SHOW DEFINITION Using the Names Only Format 6–20

Figures

1–1 Overview of DECtrace Processing 1–4
5–1 Flowchart of the Sample Application Showing One Event 5–7
A–1 The Entity Relationship Diagram for the Database A–16

Tables

2–1 Commands for Manipulating Collection Definitions 2–1
3–1 Commands for Manipulating Data Collections 3–2
4–1 Commands for Generating Reports 4–1
4–2 Format Optimization Parameters 4–3
4–3 DECtrace Reports 4–4
4–4 Data Formats 4–4
4–5 Parts of a Summary Report Event Group 4–13
5–1 DECtrace Service Routines 5–1
5–2 Resource Utilization Items 5–3
5–3 Summary of Facility Definition Options 5–13
6–1 Facility Definition Commands 6–1
6–2 Standard Resource Utilization Items 6–6
6–3 Summary of Facility Definition Options 6–20
6–4 ITEM Data Types 6–22
6–5 ITEM Default Report Widths 6–24
6–6 ITEM Usage Types 6–24
6–7 ITEM Usage Types by Data Type 6–25
7–1 DECtrace Commands 7–2
7–2 ITEM Data Types 7–11
7–3 ITEM Default Report Widths 7–13
7–4 ITEM Usage Types 7–13
7–5 ITEM Usage Types by Data Type 7–14
7–6 Rdb/VMS Optimization Parameters 7–34
8–1 DECtrace Service Routines 8–1
9–1 User Account Quotas for Using DECtrace 9–1
9–2 Troubleshooting DECtrace Problems 9–7
A–1 Rdb/VMS Representations of DECtrace-Supported Data Types . . . A–2
A–2 The point event relation EPC$1_40_REQUEST A–3
A–3 A relation for segmented strings A–4
A–4 EPC$IDENT Relation A–5
A–5 EPC$SELECTION Relation A–5

A–6 EPC$COLLECTION Relation A–6
A–7 EPC$DCF Relation A–6
A–8 EPC$REG Relation A–7
A–9 EPC$FACILITY Relation A–7
A–10 EPC$EVENT Relation A–8
A–11 EPC$ITEM Relation A–9
A–12 EPC$EVENT_ITEM Relation A–10
A–13 EPC$PROCESS Relation A–11
A–14 EPC$IMAGE Relation A–12
A–15 EPC$DCF_IMAGE Relation A–13
A–16 EPC$FACILITY_IMAGE Relation A–14
A–17 Event Relation Fields A–15
A–18 Record Type Literal A–18
A–19 FMTDICT Record A–20
A–20 COLLECTION Record A–20
A–21 FACILITY Record A–21
A–22 DCF Record A–21
A–23 IMAGE Record A–22
A–24 FACILITY-REGISTRATION Record A–23
A–25 Event Record Description A–24
A–26 Item Descriptor A–25
A–27 Data Collection Record A–26

Preface

This manual describes how to use DECtrace for VMS V1.0. The documentation refers to DECtrace for VMS by its abbreviated name: DECtrace.

Intended Audience

This manual is intended for application programmers, software performance analysts, and database administrators. Users of the DECtrace software can be divided into two distinct groups: general users and application developers. The general user will use DECtrace to gather information from existing products and applications. Developers will instrument their programs and applications with DECtrace service routine calls so that users can collect the event-based data.

General users should read this manual in the order that it is presented (Chapter 1, 2, 3, 4, 5). Application developers should read Chapter 1, then skip to Chapter 5 and Chapter 6 and then go back and read Chapters 2, 3, and 4.

System managers should read Chapter 9.

Chapters 7 and 8 provide reference material of interest to all users.

Operating System Information

Information about the versions of the operating system and related software that are compatible with this version of DECtrace is included in the DECtrace media kit in the Installation Guide.

For information on the compatibility of other software products with this version of DECtrace, refer to the System Support Addendum (SSA) that comes with the Software Product Description (SPD). You can use the SPD/SSA to verify which versions of your operating system are compatible with this version of DECtrace.

Structure

This manual has nine chapters, one appendix, a glossary, and an index:

Chapter 1	Provides an overview of the DECtrace software and describes the sample application.
Chapter 2	Describes how to create facility selections.
Chapter 3	Describes how to schedule data collection.
Chapter 4	Describes how to use the DECtrace reporting component.
Chapter 5	Describes how to instrument your application source code with DECtrace service routines.
Chapter 6	Describes how to create facility definitions.
Chapter 7	Describes the DECtrace commands.
Chapter 8	Describes the DECtrace service routines.
Chapter 9	Describes system management tasks associated with the DECtrace software.
Appendix A	Describes the formatted database and layouts produced by the DECtrace formatter.
Glossary	Defines terms used in this manual and other manuals in the documentation set.
Index	Provides page references for topics covered in this manual.

Related Documents

The other manuals in the DECtrace documentation set are:

- *DECtrace for VMS Installation Guide*—Provides instructions for installing the DECtrace software on a VMS system.
- *DECtrace for VMS Release Notes*—Provides additional information about the DECtrace software that was not included in the *DECtrace for VMS User's Guide*. The release notes are located in SYS$HELP:EPC010.RELEASE_NOTES.
- *Using DECtrace with Digital Products*—Provides product-specific information on how to use the DECtrace software with a variety of layered products.

Conventions

The special symbols used in this book are:

Symbol	Meaning
CTRL/x	This symbol tells you to press the CTRL (control) key and hold it down while pressing the specified letter key.
BOLD	Bold lettering in text indicates the definition of a new term.
[ ]	Brackets indicate optional elements.
. . .	Horizontal ellipsis indicate that you can enter additional parameters, values, or information.
. . .	Vertical ellipsis in an example means that information not directly related to the example has been omitted.
$	The dollar sign is used to indicate the DCL prompt. This prompt may be different on your system.
Color	In printed manuals, color in examples shows user input.

References to Products

The DECtrace for VMS documentation often refers to products by their abbreviated names:

- DECtrace for VMS software is referred to as DECtrace.
- VAX ACMS software is referred to as ACMS.
- VAX BLISS-32 software is referred to as BLISS-32.
- VAX C software is referred to as C.
- VAX CDD/Plus software is referred to as CDD/Plus.
- VAX COBOL software is referred to as COBOL.
- VAX DATATRIEVE software is referred to as DATATRIEVE.
- VAX FORTRAN software is referred to as FORTRAN.
- VAX Pascal software is referred to as Pascal.
- VAX Rdb/VMS software is referred to as Rdb/VMS.
- VAX SQL software is referred to as SQL.

1

Introduction to DECtrace

1.1 What Is Event-Based Data Collection?

DECtrace is a VMS layered product that collects and reports on event-based data gathered from any combination of VMS layered products and application programs containing DECtrace service routine calls. The DECtrace software is designed to operate with minimal performance impact on the system and thus can be used in both production and development environments.

DECtrace considers an **event** to be an application-defined entity. An event can have a start and an end (**duration event**), or it can simply occur (**point event**). DECtrace allows events within layered products or application programs to be defined and data items to be associated with each event. These data items can be standard resource utilization statistics (see Table 6–2) or data items specific to an application.

DECtrace differs from other collectors in that it is event based, whereas most other collectors are timer based. An event-based collector gathers data at predefined locations in your program code when that code is executed. Timer-based collectors perform data collection at specified time intervals, but at random places within your code. Advantages of event-based collectors include: (1) providing an easy way to collect and report on the actual resources utilized by certain events in applications and (2) determining the actual frequency of the execution of events, rather than an average or estimated frequency.

1.2 What Can DECtrace Do for You?

Event data collected from applications can be useful for many different purposes including:

- Tuning the performance of applications
- Planning hardware resources (capacity planning)

- Tuning the performance of databases
- Debugging applications
- Logging errors

Data can be collected from layered products or applications that already contain DECtrace calls, or you can add the calls to your own application or layered product code. If your application uses one of the already-instrumented products, called a **facility**, you can collect data automatically without adding anything to your code. Predefined facilities include ALL–IN–1, ACMS, Rdb /VMS, and VAX DBMS. See the *Using DECtrace with Digital Products* manual for descriptions of the events and items specific to these facilities.

DECtrace does not attempt to analyze or modify the performance of an application or database. Its function is to collect data requested by users and to provide reports based on that data. Interpreting these reports is the responsibility of the user or of other layered products.

1.3 How Does DECtrace Work?

When an application image that has DECtrace service routine calls in it activates, it sends a registration message to the DECtrace Registrar process saying, “Here I am, and I am ready to collect data.” DECtrace compares the information in this registration message to the criteria of any active collections on the cluster to determine if data should be collected from that process.

If a user has scheduled event data to be collected from that application (or from a facility that the application image has activated), then a “Start collecting” message is sent from DECtrace back to the image, enabling the collection of specific events and data items. The image then begins recording the appropriate event data to a data collection file until it receives a “Stop collecting” message from DECtrace or until the image terminates normally.

If no data collection is active when the image sends a registration message, it does not receive a “Start collecting” message and does not attempt to record any data. However, the DECtrace software keeps track of all processes that have registered. When a new collection begins, DECtrace compares the selection criteria against the currently registered images and sends out the appropriate “Start collecting” messages.

1.3.1 DECtrace Registrar Process

When you start the DECtrace software on your system (by using SYS$SYSTEM:EPC$STARTUP.COM), you are starting the **Registrar process**. This is a detached process that handles all communication between the applications running on your system and the DECtrace software. The Registrar tells your applications to start or stop collecting data, and maintains a list (visible using the DECtrace SHOW REGISTER command) of the processes available for data collection.

Note that a Registrar process must exist on every node from which you want to collect data. The VMS SHOW SYSTEM command confirms that the Registrar process is running on your system. For example:

```
$ SHOW SYSTEM
VAX/VMS V5.2  on node MYVAX1 11-SEP-1989 10:06:22.04   Uptime   25 23:31:29
  Pid     Process Name    State  Pri      I/O       CPU       Page flts Ph.Mem
25200021 SWAPPER         HIB     16        0   0 00:33:18.22         0      0
25200062 SMITH           LEF      4     1132   0 05:17:49.86     23591    512
25200026 ERRFMT          HIB      8    19371   0 00:05:19.66        76     97
25200027 CACHE_SERVER    HIB     16      302   0 00:00:01.61        63     90
25200028 CLUSTER_SERVER  HIB     10     1308   0 00:00:40.74       124    208
25200029 OPCOM           HIB      8     2847   0 00:06:34.99       749    134
2520002A JOB_CONTROL     HIB     10    22829   0 00:06:23.35       258    445
2520002B CONFIGURE       HIB      8       18   0 00:00:00.43       104    148
2520022D SYSTEM          CUR      4   115996   0 00:09:17.22     71090    258
25200030 NETACP          HIB     10     2711   0 00:19:15.60     79608   1500
25200032 REMACP          HIB      9       19   0 00:00:00.26        75     51
25200338 SMITH_1         LEF      6    14262   0 00:03:16.16     37920    512
252001DA EPC$REGISTRAR   HIB      8    10955   0 00:04:36.88     21334    512
25200588 RDMS_MONITOR    LEF     15      558   0 00:00:02.29      2720     53
```

1.3.2 DECtrace Administration Database

When you install the DECtrace software on your system or VAXcluster, the installation procedure creates an Rdb/VMS database that DECtrace uses to store the following:

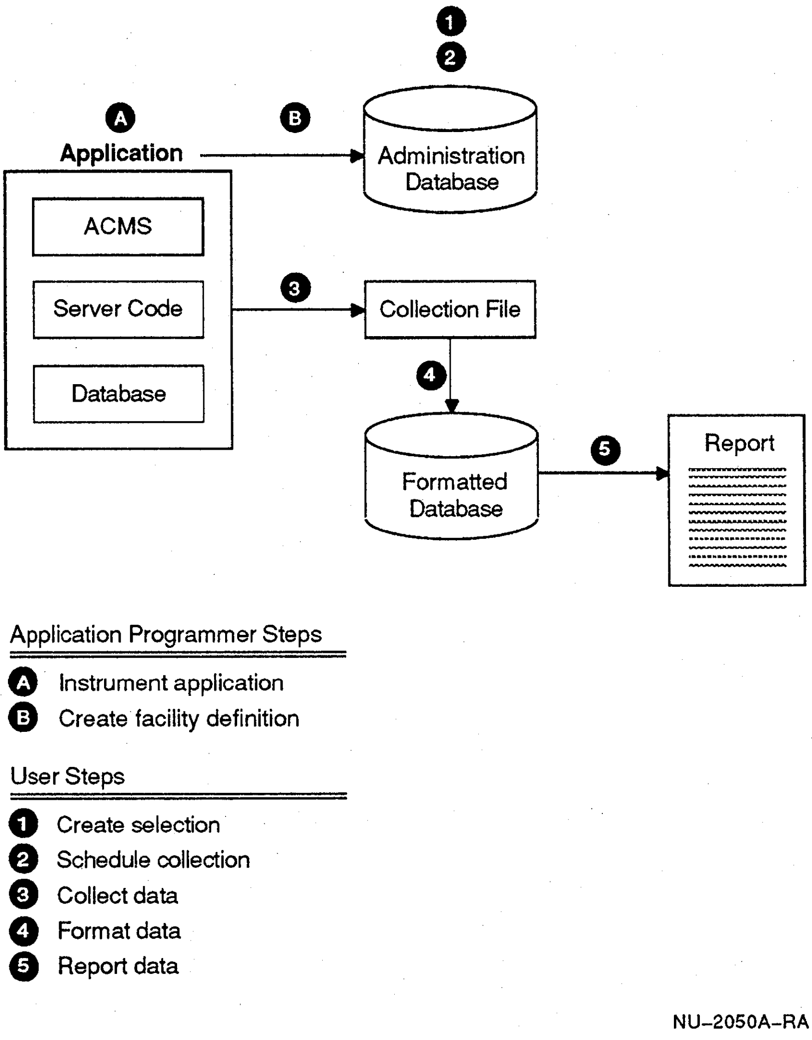
- Facility definitions
- Facility selections
- Data collection schedule information

The database is referred to as the **DECtrace administration database**. A system-wide logical name, EPC$ADMIN_DB, points to the location of the database.

1.4 Overview of DECtrace Processing

Figure 1–1 illustrates the process of using the DECtrace software to collect and report on event-based data.

Figure 1–1 Overview of DECtrace Processing



Collecting data from predefined facilities consists of the following steps:

❶ User creates a facility selection, naming the facilities from which to collect data.

❷ User schedules data collection referencing the facility selection created in Step 1 (or, a preexisting selection).

❸ Someone runs the application program or layered product, causing predefined events to execute and be collected by DECtrace.

❹ User formats data collection files into either an Rdb/VMS database or an RMS file.[1]

❺ User generates a DECtrace report from the formatted Rdb/VMS database or uses a report generator (for example, VAX DATATRIEVE) to create a report from either the Rdb/VMS database or RMS file.

You can also use DECtrace to collect event data from your own applications. Creating your own facility consists of the following steps (then follow the general user steps outlined in the previous section):

A Application programmer instruments the source code with DECtrace service routine calls.

B Application programmer creates a facility definition for the product and stores it in the DECtrace administration database.

1.5 Explanation of Sample Application

For application programmers who want to instrument their own code with DECtrace service routine calls, a sample application that uses DECtrace is included with the DECtrace software kit. The application simulates a simple bank automated teller machine (ATM). Duration events are defined for each of the basic transactions: checking a balance, depositing funds, and withdrawing funds. A point event is defined to note execution of the error handling procedure.

The instrumentation of the ATM application is designed to gather information about the user interface. The goal is to be able to answer questions such as:

- How long does it take to complete a transaction?
- Do customers check their balance before or after every transaction?
- Could overdrafts be reduced or eliminated by displaying the balance on the withdrawal display?
- Are ATM machines used primarily for deposits or withdrawals?

[1] The layouts of the Rdb/VMS and RMS file formats are described in Appendix A.

Various examples throughout this manual are based on the VAX COBOL version of the sample application. The source file for this program, along with versions written in VAX FORTRAN and VAX Pascal, are located in EPC$EXAMPLES. The three versions perform the same functions and collect the same event data. To use the sample application, copy the executable (EPC$ATM-SAMPLE.EXE) and the data file (EPC$ATM-SAMPLE.DAT) to your directory. The data file contains ten account records, numbered from 1 to 10.

2 Creating a Facility Selection

A **facility selection** describes which data to collect during data collection. It specifies the individual facilities (applications and layered products) and the **class** of data to collect for each one. Creating a facility selection is the first step (see Figure 1–1) for general users of the DECtrace software who want to gather event-based data from facilities on their system. See Chapter 5 and Chapter 6 for information on how to create a new facility.

This chapter describes how to manipulate facility selections. It describes how to:

- Create selections
- Delete selections
- Display selections

Table 2–1 summarizes the DECtrace commands available to accomplish the functions associated with facility selections.

Table 2–1 Commands for Manipulating Collection Definitions

Command	Description
CREATE SELECTION	Creates a facility selection in the DECtrace administration database. Selection names must be unique in the DECtrace administration database. If you use the /REPLACE qualifier, your new facility selection replaces the old selection of the same name.
DELETE SELECTION	Deletes a facility selection from the DECtrace administration database.
SHOW SELECTION	Displays a facility selection in one of three formats: BRIEF, FULL, or NAMES_ONLY.

2.1 Choosing Which Data to Collect

The DECtrace software performs data collection on your system with minimal performance impact to the application and the system. However, the impact increases as you select more facilities from which to collect data, and as you specify more data to collect from each facility. The data files produced by DECtrace can become very large in a busy environment. If you are on a system with limited free disk space, you should carefully consider how much data you will be collecting. If you use DECtrace to collect all of the available data from all of the facilities on your system, you could run out of disk space very quickly.

When you decide to use DECtrace you probably have a specific function or area about which you want more information. For example, some of your applications might not be performing as well as you think they should, or you might need to generate statistics for a capacity planning report for the next fiscal year. DECtrace can collect a wide variety of data from the applications running on your system. However, you may not need to collect all of the possible data from all of the events occurring in the facilities you are using. It is possible to end up with much more information than you really need.

To make it easier to collect the data that you really need and to protect you from collecting data that you do not need, facilities can have one or more **collection classes** associated with them. These classes are subsets of all of the available events and items chosen by the facility developers for their significance to a specific function. For example, a facility might have specific collection classes defined for performance, workload analysis, capacity planning, or debugging purposes. The importance of selecting a collection class is illustrated by the following collection rates, which apply to the Debit/Credit workload for Rdb/VMS and VAX DBMS:

- 20,000+ blocks per 1 TPS/hour for ALL class data
- 20,000 blocks per 1 TPS/hour for PERFORMANCE class data
- 10,000 blocks per 1 TPS/hour for WORKLOAD class data

You use the SHOW DEFINITION/FORMAT=NAMES_ONLY command to determine what classes are available for a given facility. For example, Example 2–1 shows that the NEW_FORMS facility has the collection classes ALL, CAPACITY_PLANNING, PERFORMANCE, and WORKLOAD (default). ATM_SAMPLE and MY_APPLICATION have only the default (ALL) class available.

Example 2–1 Display Format for SHOW DEFINITION /FORMAT=NAMES_ONLY

```
23-DEC-1989 11:43       Facility Definition Information             Page 1
Names Only Report                                           DECtrace V1.0-0

 Facility:               Version:    Creation Date:      Class:
 ---------------------   ----------  ------------------  ------------------
 ATM_SAMPLE              V1.0        25-AUG-1989 14:17   ALL                (D)
 MY_FACILITY             V2.0        12-AUG-1989 10:27   ALL                (D)
 NEW_FORMS               V4.0        13-JUN-1989 07:22   ALL
                                                         CAPACITY_PLANNING
                                                         PERFORMANCE
                                                         WORKLOAD           (D)
 .
 .
 .
```

Every facility has the ALL class, which contains all of the events and items defined for the facility. Furthermore, one of the classes is designated by the facility developers as the default class based on its predicted importance and frequency of use. Note that unless otherwise specified, the default is the ALL class, which is generated automatically when the developer creates a facility definition.

2.2 Creating a Selection

A DECtrace facility selection consists of:

- Name of the selection
- List of facilities from which to collect data
- Classes of data to collect for each facility
- Comment describing the purpose of the selection

The format of the CREATE SELECTION command is:

```
                                  [ /FACILITY=(facility_name[, . . . ]) ]
CREATE SELECTION selection_name   [ /COMMENT=" . . . "                  ]
                                  [ /OPTIONS[=file_spec]                ]
                                  [ /REPLACE                            ]
```

All facility selections are stored in the DECtrace administration database, and any user can reference a facility definition created by any other user. The facility selections remain in the database until they are deleted by their creator or by a user with BYPASS or SYSPRV privilege.

You use the CREATE SELECTION command to choose a subset of the available facilities from which you want to collect data. The following example defines the facility selection MY_SELECTION to collect the default class of data for ACMS:

```
$ COLLECT CREATE SELECTION MY_SELECTION /FACILITY=ACMS -
_$  /COMMENT="Collect the default VAX ACMS data"
```

To define a more detailed facility selection, you should use an options file. The /OPTIONS qualifier allows you to specify more than one facility and choose a different class of data for each facility. The qualifier takes the name of an options file as an argument. If you do not specify a file name, DECtrace prompts you for the options. Each facility must be defined on a separate line. The format of the facility description is:

FACILITY facility-name *[/VERSION="version-code"] [/CLASS=class-name]*

facility-name
The name of the facility from which to collect data.

version-code
A text string identifying the version of the facility. The string must be enclosed with quotation marks (" "). If you do not include the version code, DECtrace uses the most recent version of the facility registered in the DECtrace administration database.

class-name
The name of the class of data to collect for the facility.

The following example creates the facility selection COLLECT_ALL to collect all of the possible events and data items for MY_APPLICATION and the performance events and items for Rdb/VMS:

```
$ COLLECT CREATE SELECTION COLLECT_ALL /OPTIONS
Option> FACILITY MY_FACILITY /CLASS=ALL
Option> FACILITY RDBVMS /CLASS=PERFORMANCE
Option> CTRL/Z
%EPC-S-SELCRE, Selection COLLECT_ALL was created
$
```

2.3 Deleting a Selection

You can delete a facility selection from the DECtrace administration database with the DELETE SELECTION command. You must be the creator of the selection or have VMS BYPASS or SYSPRV privilege. Note that you cannot delete a facility selection if any data collection, either active or pending, is using that facility selection.

The following example deletes the facility selection TEMP_SELECTION:

```
$ COLLECT DELETE SELECTION TEMP_SELECTION /NOCONFIRM
%EPC-S-SELDEL_DELETED, Selection TEMP_SELECTION was deleted
$
```

To delete a facility selection that is referenced by any active or pending collections, you must first cancel the data collection. See Section 3.4 for information on how to cancel all data collection using a particular facility selection.

You can use the SHOW SELECTION/FORMAT=NAMES_ONLY command to confirm the spelling of the names of facility selections that you want to delete.

2.4 Displaying Information About a Selection

You can display information about the facility selections stored in the DECtrace administration database using the SHOW SELECTION command. The command takes one argument: the name of a selection. If you do not specify a selection name, DECtrace displays information on all of the facility selections defined on the system.

You can specify the amount of information to display about a selection by using the /FORMAT qualifier to the SHOW SELECTION command. There are three valid format types:

- BRIEF (default)
- FULL
- NAMES_ONLY

2.4.1 BRIEF Format

If you specify /FORMAT=BRIEF with the SHOW SELECTION command, DECtrace lists the names of the facility selections in the DECtrace administration database together with the facilities, versions, and collection classes that each selection describes. If you did not specify a facility version when you created the facility selection, the most recently created version of the facility definition is used and "(latest)" is displayed.

The BRIEF format is useful if you want to check what facilities are in a facility selection. This will also show you if there is already a selection defined that suits your collection needs.

Example 2–2 shows a sample of the display produced with the SHOW SELECTION /FORMAT=BRIEF command.

Example 2–2 Display for SHOW SELECTION Using the Brief Format

```
9-MAY-1989 10:03          Facility Selection Information               Page 1
                                                                 DECtrace V1.0

   Selection Name         Facility          Version      Class
   --------------------   ---------------   ----------   --------------------
   A1_DATA_ENTRY          OA                (latest)     ALL
                          TESTER            T4.1         ALL
                          RDBVMS            V3.1         ALL

   ACMSDBMS               ACMS              (latest)     PERFORMANCE
                          DBMS              (latest)     PERFORMANCE

   RDB_LOAD_TEST          RDBVMS            V3.1         WORKLOAD
```

2.4.2 FULL Format

If you specify /FORMAT=FULL with the SHOW SELECTION command, DECtrace displays a full description of facility selections stored in the DECtrace administration database. You can display the description of a single selection if you include its name on the command line. For example, the following command would display the complete description of the FINANCE selection:

```
$ COLLECT SHOW SELECTION FINANCE /FORMAT=FULL
```

The FULL format display includes the following information:

- Name of the facility selection
- User name of the facility selection's creator
- Facility name, version, and collection class for each facility specified in the selection

Example 2–3 shows a sample of the display produced with the SHOW SELECTION /FORMAT=FULL command.

Example 2–3 Display for SHOW SELECTION Using the Full Format

```
9-MAY-1989 10:04         Facility Selection Information             Page 1
                                                              DECtrace V1.0

   Selection:           ACMSDBMS
   Comment:             This is the facility selection for VAX ACMS
                        and VAX DBMS performance data.
   Created By:          JONES

   Facility:            ACMS
   Version:             (latest)
   Collection class:    PERFORMANCE

   Facility:            DBMS
   Version:             (latest)
   Collection class:    PERFORMANCE
```

2.4.3 NAMES_ONLY Format

If you specify /FORMAT=NAMES_ONLY with the SHOW SELECTION command, DECtrace lists the names of the facility selections alphabetically. This format is useful to determine the correct spelling of a selection name or to determine if a particular selection name already exists in the DECtrace administration database.

Example 2–4 shows a sample of the display produced with the SHOW SELECTION /FORMAT=NAMES_ONLY command.

Example 2–4 Display for SHOW SELECTION Using the Names Only Format

```
9-MAY-1989 10:03         Facility Selection Information             Page 1
                                                              DECtrace V1.0

   Selection Name
   --------------------------------
   A1_DATA_ENTRY
   ACMSDBMS
   RDB_LOAD_TEST
```

3 Scheduling Data Collection

Scheduling data collection is the second step (see Figure 1–1) for general users of the DECtrace software. **Data collection** is the process of gathering event-based data from facilities and applications running on your system. You define the process by means of the SCHEDULE COLLECTION command which allows you to specify:

- Which data to collect
- How much data to collect
- When to collect data
- Where to store collected data

This chapter describes how to work with DECtrace data collections including:

- Scheduling data collection
- Canceling data collection
- Displaying information about collections scheduled in the DECtrace administration database

Table 3–1 summarizes the DECtrace commands available to accomplish the functions associated with DECtrace data collection.

Table 3–1 Commands for Manipulating Data Collections

Command	Description
SHOW REGISTER	Shows the individual processes for which data can be collected.
SCHEDULE COLLECTION	Schedules data collection based on the specified qualifiers.
CANCEL COLLECTION	Stops data collection for an active collection. If a collection is pending, it removes the collection from the schedule.
SHOW COLLECTION	Shows data collection information in one of two formats, either BRIEF or FULL.
SHOW HISTORY	Shows all error or informational messages that have occurred during one or all data collections active on your system.

3.1 Choosing the Processes from Which to Collect Data

The amount of data collected during data collection is dependent on how many processes on each node are collecting data. By default, DECtrace gathers data from all processes that are using the facilities specified in your facility selection. You can optionally select a subset of the processes that would normally collect data.

For example, if you have two processes on your system which are running different ACMS applications, you can choose to collect data from either process or from both.

3.1.1 Process Registration

When an image that contains DECtrace service routine calls activates, it issues a call to EPC$INIT which registers both the process and the facility with DECtrace. DECtrace maintains a user-visible register of information about these images and processes. The register information includes:

- Name of the image
- Name of the facilities that issued the DECtrace initialization information
- Image-specific registration identifier(s) (ID) for the process
- User name, process name, and process identification number (EPID) of the process in which the facility is running
- Node on which the process is running

Several facilities may register for each process. For example, a program named TEST_PROG may use both Rdb/VMS and VAX DBMS. In this case, each time a user runs TEST_PROG.EXE, that user's process will register both facilities with DECtrace. The facilities remain registered until the image terminates. Note that it is not necessary for TEST_PROG to contain any DECtrace service routine calls. DECtrace collects the predefined events and items in Rdb/VMS and VAX DBMS as they occur as part of TEST_PROG's execution.

3.1.2 Using the Registration ID

A **registration ID** is an optional facility-specific character string that is known to the facility only at run time. When a facility is invoked, it can optionally pass a registration ID to the DECtrace Registrar by means of the EPC$INIT service routine. The registration ID is useful in distinguishing separate images that are using the same facilities. For example, if there are two processes on your system running different ACMS applications, you can use the registration ID to collect data from only one of them. ACMS provides the ACMS application name as the registration ID.

DECtrace also registers process-specific information when an image containing DECtrace service routine calls activates. The process-specific registration IDs include the EPID, the image name, the user name, and the process name.

You use the /REGISTRATION_ID qualifier on the SCHEDULE COLLECTION command to collect data from processes with a specific registration ID. This qualifier is not a substitute for a facility selection. It provides additional restrictions to what you will be collecting.

You can specify one or more registration IDs when you schedule data collection. This is useful for collecting data from only a few of the processes that are using the facilities listed in your facility selection. Only those processes whose registration IDs match the specified values (and also contain at least one facility specified in your facility selection) will collect data when data collection begins. This allows you to greatly reduce the scope of your collection. For example, the following command schedules a collection to gather data from one user (WEBSTER) running a specific application (SCHEDULE.EXE):

```
$  COLLECT SCHEDULE COLLECTION COURSE_SCHEDULE DANIEL.DAT-
_$  /SELECTION=JUST_RDB /NOCLUSTER -
_$  /BEGINNING=09:00 /ENDING=11:30 -
_$  /REGISTRATION_ID=(WEBSTER,WORK1:[TOOLS]SCHEDULE.EXE)
_$  /PROTECTION=(G:W)
%EPC_S_SCHED, Data collection COURSE_SCHEDULE is scheduled
```

See Section 3.3.1 for more information on collecting per-user data.

The following example tells DECtrace to select those processes on the cluster that have ORDER_ENTRY as a registration ID. Then DECtrace collects the ACMS and Rdb/VMS data from those processes:

```
$  COLLECT SCHEDULE COLLECTION ORDER_WORK ORDERS.DAT -
_$  /SELECTION=ACMS_AND_RDB /CLUSTER -
_$  /BEGINNING=09:00 /ENDING=10:00 -
_$  /REGISTRATION_ID=ORDER_ENTRY
%EPC_S_SCHED, Data collection ORDER_WORK is scheduled
```

Example 3–1 shows the DECtrace register after the collection ORDER_WORK activates. ACMS and Rdb/VMS event data is collected from the following processes on MYVAX1: SMITH_2 and WRITER and from the following processes on MYVAX2: CONTRACT and JONES_2. Note that SMITH_2 also registered the MY_FACILITY facility. Because MY_FACILITY is not included in the facility selection, DECtrace does not collect data from it. In addition, note that SECRETARY registered both ACMS and Rdb/VMS. However, the process does not have the correct registration ID, so no data is collected from it.

Example 3–1 Process Registration Display

```
12-JUN-1989 9:03:22.4         Register Information for Cluster        Page 1
                                                                DECtrace V1.0-0

Registrations not collecting

  Node:  MYVAX1

    Process   Process Name     Facility         Version   Registration ID
    --------  ---------------  ---------------  --------  --------------------
    21444556  JONES            RDBVMS           V3.1
    WORK$DISK:[FINANCE]DEBIT_CREDIT.EXE;11

    21544675  SMITH            RDBVMS           V3.1
    USER2:[GAMES]POKER.EXE;3

    21457890  SMITH_2          DBMS             V4.1
                               MY_FACILITY      V2.0      DATA_ENTRY
    WORK$DISK:[TOOLS]INVENTORY_CHECK.EXE;5

  Node:  MYVAX2

    Process   Process Name     Facility         Version   Registration ID
    --------  ---------------  ---------------  --------  --------------------
    21778900  SECRETARY        ACMS             V3.1      BUDGET_ANALYSIS
                               RDBVMS           V3.1
                               MY_FACILITY      V2.0      DATA_ENTRY
    WORK$DISK:[FINANCE]BUDGET_UPDATE.EXE;24

    21778902  JONES            ACMS             V3.1      BUDGET_ANALYSIS
    WORK$DISK:[FINANCE]DEBIT_CREDIT.EXE;11

Registrations actively collecting
```

(continued on next page)

Example 3–1 (Cont.) Process Registration Display

```
Node:  MYVAX1        Collection:  ORDER_WORK     Selection:  ACMS_AND_RDB

  Process    Process Name     Facility         Version    Registration ID
  --------   ---------------  ---------------  --------   -------------------
  21457890   SMITH_2          ACMS             V3.1       ORDER_ENTRY
  WORK$DISK:[TOOLS]INVENTORY_CHECK.EXE;5

  21567444   WRITER           ACMS             V3.1       ORDER_ENTRY
  WORK$DISK:[TOOLS]SPELL_CHECK.EXE;5


Node:  MYVAX2        Collection:  ORDER_WORK     Selection:  ACMS_AND_RDB

  Process    Process Name     Facility         Version    Registration ID
  --------   ---------------  ---------------  --------   -------------------

  21778888   CONTRACT         ACMS             V3.1       ORDER_ENTRY
                              RDBVMS           V3.1
  WORK$DISK:[TOOLS]INVENTORY_CHECK.EXE;5

  21778905   JONES_2          ACMS             V3.1       ORDER_ENTRY
                              RDBVMS           V3.1
  USER1:[JONES.TESTS]ORDER_ENTRY_PROTO.EXE;1
```

3.1.3 Displaying Information about Process Registration

The SHOW REGISTER command allows you to examine the current state of process registration on your system or VAXcluster. The SHOW REGISTER display is divided into the following segments:

- Registrations with no facility definitions

  These processes actually represent an error condition where a facility has made a call to EPC$INIT, but there is no corresponding facility definition in the DECtrace administration database. No data can be collected for an undefined facility.

- Registrations not collecting

  These processes do not have any active collections collecting data from them, but they are available for data collection if the proper collection is scheduled.

- Registrations actively collecting

  These processes have active data collection occurring.

Example 3–2 shows a typical register display produced by the SHOW REGISTER/CLUSTER command. In the example, three separate processes are running the WEEKLY_CHECKS program which has registered three facilities: Rdb/VMS and two facilities for which no facility definitions exist in the DECtrace administration database. Another process is running the INVESTMENTS program which registered two facilities: Rdb/VMS and NEW_FORMS. Lastly, the MORGAN process is running the MAINT_EMP program which also registered Rdb/VMS and NEW_FORMS, but has a different registration ID than the INVESTMENTS program. Note that the registration ID is optional, and not all facilities choose to provide one.

Two local collections are active on the cluster: ALL_DAY is active on MYVAX1, and WED_2ND_SHIFT is active on MYVAX2.

Example 3–2 Display Format for SHOW REGISTER/CLUSTER

```
2-AUG-1989 21:25        Register Information for Cluster               Page 1
                                                              DECtrace V1.0-0

Registrations with no Facility definitions

  Node: MYVAX1

    Process   Process Name     Facility  Registration Id
    --------  ---------------  --------  ---------------------------------------
    0000004D  Wed Payroll          2235  Payroll
                                   2236  Stock Plan
    DISK$PAYROLL:[PAYROLL.IMAGES]WEEKLY_CHECKS.EXE;80

  Node: MYVAX2

    Process   Process Name     Facility  Registration Id
    --------  ---------------  --------  ---------------------------------------
    00000021  Monday Payroll       2235  Payroll
                                   2236  Stock Plan
    DISK$PAYROLL:[PAYROLL.IMAGES]WEEKLY_CHECKS.EXE;80

    00000025  Tuesday Payroll      2235  Payroll
                                   2236  Stock Plan
    DISK$PAYROLL:[PAYROLL.IMAGES]WEEKLY_CHECKS.EXE;80

Registrations not collecting

  Node: MYVAX1

    Process   Process Name     Facility  Version   Registration Id
    --------  ---------------  --------  --------  ----------------------------
    00000021  Monday Payroll   RDBVMS    V3.1

    DISK$PAYROLL:[PAYROLL.IMAGES]WEEKLY_CHECKS.EXE;80

    00000025  Tuesday Payroll  RDBVMS    V3.1

    DISK$PAYROLL:[PAYROLL.IMAGES]WEEKLY_CHECKS.EXE;80

    00000027  MORGAN           RDBVMS    V3.1

    DISK$PAYROLL:[PAYROLL.IMAGES]MAINT_EMP.EXE;3
```

(continued on next page)

Example 3–2 (Cont.) Display Format for SHOW REGISTER/CLUSTER

```
Registrations actively collecting

  Node: MYVAX1         Collection: ALL_DAY     Selection: FORMS_AND_DB

    Process   Process Name     Facility  Version   Registration Id
    --------  ---------------  --------  --------  ---------------------------
    00000059  LINDA            NEW_FORM  T1.0-1    Investment Stream
                               RDBVMS    V3.1

    DISK$PAYROLL:[PAYROLL.IMAGES]INVESTMENTS.EXE;3

    0000004D  Wed Payroll      RDBVMS    V3.1
    DISK$PAYROLL:[PAYROLL.IMAGES]WEEKLY_CHECKS.EXE;80


2-AUG-1989 21:25        Register Information for Cluster            Page 2
                                                           DECtrace V1.0-0

Registrations actively collecting

  Node: MYVAX2  Collection: WED_2ND_SHIFT     Selection: JUST_FORMS

    Process   Process Name     Facility  Version   Registration Id
    --------  ---------------  --------  --------  ---------------------------
    00000027  MORGAN           NEW_FORM  T1.0-1    Payroll Stream
    DISK$PAYROLL:[PAYROLL.IMAGES]MAINT_EMP.EXE;3
```

Example 3–3 shows the processes registered on the local node (MYVAX2).

Example 3–3 Display Format for SHOW REGISTER/NOCLUSTER

```
2-AUG-1989 21:23        Register Information for node MYVAX2        Page 1
                                                           DECtrace V1.0-0

Registrations with no Facility definitions

  Node: MYVAX2

    Process   Process Name     Facility  Registration Id
    --------  ---------------  --------  --------------------------------------
    00000021  Monday Payroll       2235  Payroll
                                   2236  Stock Plan
    DISK$PAYROLL:[PAYROLL.IMAGES]WEEKLY_CHECKS.EXE;80

    00000025  Tuesday Payroll      2235  Payroll
                                   2236  Stock Plan
    DISK$PAYROLL:[PAYROLL.IMAGES]WEEKLY_CHECKS.EXE;80
```

(continued on next page)

Example 3–3 (Cont.) Display Format for SHOW REGISTER/NOCLUSTER

```
Registrations not collecting

  Node: MYVAX2

    Process   Process Name     Facility  Version   Registration Id
    --------  ---------------  --------  --------  ---------------------------
    00000021  Monday Payroll   RDBVMS    V3.1
    DISK$PAYROLL:[PAYROLL.IMAGES]WEEKLY_CHECKS.EXE;80
    00000025  Tuesday Payroll  RDBVMS    V3.1
    DISK$PAYROLL:[PAYROLL.IMAGES]WEEKLY_CHECKS.EXE;80
    00000027  MORGAN           RDBVMS    V3.1
    DISK$PAYROLL:[PAYROLL.IMAGES]MAINT_EMP.EXE;3

Registrations actively collecting

  Node: MYVAX2  Collection: WED_2ND_SHIFT    Selection: JUST_FORMS

    Process   Process Name     Facility  Version   Registration Id
    --------  ---------------  --------  --------  ---------------------------
    00000027  MORGAN           NEW_FORM  T1.0-1    Payroll Stream
    DISK$PAYROLL:[PAYROLL.IMAGES]MAINT_EMP.EXE;3
```

3.2 Scheduling a Collection

You must schedule data collection on your system before DECtrace can begin gathering data. Data collection criteria include the output file for the collected data, the start and end times (or alternately, the duration), which facility selection to use, and whether to collect from your entire cluster or just the local node. Note that although you can schedule many collections on a node, only one data collection can be active on a node at any time. The DECtrace software does not allow you to schedule collections that overlap or run simultaneously. DECtrace detects the conflict immediately when you attempt to schedule the conflicting collection.

You can specify data collection to occur either locally or cluster-wide using the /[NO]CLUSTER qualifier. By default, SCHEDULE COLLECTION schedules data collection to occur on every node in the cluster. To schedule a collection on a subset of the cluster, you must log in to each node that you want data collection to occur on and schedule a local collection on that node. Note that on a standalone system, the /CLUSTER qualifier is ignored.

When you successfully schedule data collection, and again when the collection activates, DECtrace writes a confirmation message to the history database. You can examine the history database at any time with the SHOW HISTORY command. See Section 3.5.2 for information about the history database and the SHOW HISTORY command.

3.2.1 Scheduling Data Collection on a Standalone System

If you have DECtrace installed on a standalone system, you can collect data from applications running on that system.

The following example schedules the collection MY_TEST to begin at 11:00 and end at 12:00 on the current day. The collection uses the facility selection MY_SELECTION and runs on the local node. DECtrace stores the collected data in the file MY_DATA.DAT in your default device and directory:

```
$  COLLECT SCHEDULE COLLECTION MY_TEST MY_DATA.DAT -
_$  /SELECTION=MY_SELECTION -
_$  /BEGINNING=11:00 /ENDING=12:00
%EPC-S-SCHED, Data collection MY_TEST is scheduled
```

Alternately, you can use the /DURATION qualifier in place of the /ENDING qualifier. You must specify the duration as a relative VMS time. For example:

```
$  COLLECT SCHEDULE COLLECTION MY_TEST MY_DATA.DAT -
_$  /SELECTION=MY_SELECTION -
_$  /BEGINNING=11:00 /DURATION="1:" -
_$  /NOCLUSTER
%EPC-S-SCHED, Data collection MY_TEST is scheduled
```

3.2.2 Scheduling Data Collection on a Cluster

If you run DECtrace on a system that is a member of a VAXcluster, your collections are automatically scheduled to run on all nodes in the cluster.

The following example schedules the collection MY_FULL_TEST to begin at 10:00 and end at 11:00 on the current day. The collection uses the facility selection MY_SELECTION and runs on every node in the cluster. DECtrace stores the collected data in the file CLUSTER_DATA.DAT in your default device and directory:

```
$  COLLECT SCHEDULE COLLECTION MY_FULL_TEST CLUSTER_DATA.DAT -
_$  /SELECTION=MY_SELECTION -
_$  /BEGINNING=10:00 /ENDING=11:00
_$  /PROTECTION=(W:W)
%EPC-S-SCHED, Data collection MY_FULL_TEST is scheduled
```

3.2.3 Scheduling Data Collection on Part of a Cluster

If you run DECtrace on a VAXcluster but you do not want to collect data from every node, use the /NOCLUSTER qualifier when you schedule your data collection. This will schedule a **local collection** (data collection that occurs only on your local node).

The following example schedules the collection MY_LOCAL_TEST to begin at 11:00 and end at 12:00 on the current day. The collection uses the facility selection MY_SELECTION and runs on the local node (that is, the node you are currently logged in to). DECtrace stores the collected data in the file LOCAL_DATA.DAT in your default device and directory:

```
$ COLLECT SCHEDULE COLLECTION MY_LOCAL_TEST LOCAL_DATA.DAT -
_$  /SELECTION=MY_SELECTION -
_$  /BEGINNING=11:00 /ENDING=12:00 -
_$  /NOCLUSTER
%EPC-S-SCHED, Data collection MY_LOCAL_TEST is scheduled
```

To schedule data collection on a subset of the cluster, you must log in to each node that you want data collection to occur on and schedule a local collection on that node. Note that you should use a different name for the data files produced by each of these collections. One suggestion is to include the node name as part of the data file name. If you do not use different names for the data files, DECtrace uses the file version number to distinguish each file. The individual data files can be combined later using the FORMAT command. See Section 4.2 for information on combining multiple data files.

Example 3–4 shows a sample command procedure that schedules a local collection with the node name included as part of both the collection and data file names.

Example 3–4 Sample Command Procedure for Starting a Local Collection

```
$!
$!                       LOCAL_COLLECTION.COM
$!
$! This procedure starts DECtrace data collection using the
$! node name as part of both the collection name and the data file
$! name.
$!
$! Find the name of the local node and append it to the collection
$! and data file names
$!
$ node_name  = F$GETSYI("NODENAME")
$ collection_name  = "MY_COLL_''node_name'"
$ data_file  = "DATA_''node_name'.DAT"
$!
$! Put your facility selection name and start and end times here:
$!
$ selection_name = "MY_SELECTION"
$ start_time = "12:00"
$ end_time   = "13:00"
$!
$! Schedule your local data collection
$!
$ COLLECT SCHEDULE COLLECTION  'collection_name' 'data_file' -
     /SELECTION='selection_name' -
     /BEGIN='start_time' /END='end_time' -
     /NOCLUSTER /PROTECTION=(W:W)
$ EXIT
```

If you have the VMS OPER privilege, you can use the VMS System Management (SYSMAN) utility to schedule local data collection on nodes in your cluster without logging in to each node.[1] SYSMAN is a utility that centralizes the management of nodes and clusters by allowing you to define an environment that can be a particular node, a group of nodes, or a cluster. You can perform tasks on all nodes in the environment from your local node. The following example shows how you can schedule local data collection on nodes MYVAX1 and MYVAX2 from your local node.

```
$  RUN SYS$SYSTEM:SYSMAN
SYSMAN>  SET ENVIRONMENT/NODE=(MYVAX1,MYVAX2)
%SYSMAN-I-ENV, Current Command Environment:
        Individual nodes: MYVAX1,MYVAX2
        Username SMITH will be used on nonlocal nodes
SYSMAN>  DO @LOCAL_COLLECTION.COM
%SYSMAN-I-OUTPUT, Command execution on node MYVAX1
%SYSMAN-I-OUTPUT, Command execution on node MYVAX2
SYSMAN>  EXIT
$
```

3.3 Data Collection Files

When you schedule data collection, you can specify one or many data files to store the event data (refer to the SCHEDULE COLLECTION command in Chapter 7 for details). In addition, one or many different executable images (running in the context of one or many processes) that are collecting data on the local system or VAXcluster can record the data in a common data collection file. By default, there is one data collection file for all of the data related to a particular collection. However, if the amount of data collected is too great for the I/O bandwidth of the disk where the data file exists, you can specify more than one file when scheduling data collection. In this case, each data file would be on a different disk device. When you specify more than one file, DECtrace performs a round-robin style load balancing among all executable images that collect data. See Section 4.2 for information on how to combine these multiple data files after data collection has ended.

The following example schedules data collection using three data collection files, each on a separate device:

```
$  COLLECT SCHEDULE COLLECTION TSTVAX_WORKLOAD -
_$  USER1:[SMITH.DATA]TST1.DAT,USER2:[DATA]TST2.DAT, -
_$  USER3:[DATA]TST3.DAT /SELECTION=ACMS_AND_RDB -
_$  /BEGIN=09:00 /END=17:00 /CLUSTER
```

[1] If OPER is not a default privilege for your process, you must use the SET PROFILE /PRIVILEGE=OPER command in SYSMAN. Refer to the *VMS SYSMAN Utility Manual* for information about the SYSMAN utility.

Alternately, you could use a **file list**, which is a file containing a list of file specifications to use as output files. Note that the /FILELIST qualifier is position-dependent; you must specify it on the second parameter to the command. For example:

```
$ COLLECT SCHEDULE COLLECTION TSTVAX_WORKLOAD -
_$ DATA_LIST.TXT /FILELIST -
_$ /SELECTION=ACMS_AND_RDB /BEGIN=09:00 /END=17:00 /CLUSTER
```

Each file specification must be on a separate line within the file list. For example:

```
USER1:[SMITH.DATA]TST1.DAT
USER2:[DATA]TST2.DAT
USER3:[DATA]TST3.DAT
```

3.3.1 File Protection Schemes

DECtrace uses the default file protection for your process when creating the data collection file(s) for a collection. For a process to record event data to your file, it must have write access to that file. Use the /PROTECTION qualifier to the SCHEDULE COLLECTION command to automatically set the protection on your data collection files. Note that the data collection files are created immediately when the collection is scheduled, not when the collection becomes active.

The format of the qualifier is:

/PROTECTION=(ownership:access[, . . .])

Where:

- OWNERSHIP is one of the following: (S)ystem, (O)wner, (G)roup, or (W)orld.
- ACCESS is any combination of the following: (R)ead, (W)rite, (E)xecute, or (D)elete.

If you do not specify a value for each ownership category, or if you omit the /PROTECTION qualifier, the DECtrace software applies the current default protection for each unspecified category. If the data collection file replaces a previous version, then the protection on the old file is used on the new one.

The following example schedules a collection where all members of the user's UIC-based group have write access to the data collection file:

```
$ COLLECT SCHEDULE COLLECTION ALL_DAY MONDAY.DAT -
_$ /SELECTION=EVERYTHING /BEGIN=9:00 /END=17:00 -
_$ /PROTECTION=(G:W)
```

3.4 Canceling Data Collection

You can cancel data collection that is active or pending with the CANCEL COLLECTION command. When you cancel active data collection, DECtrace stops recording data for the collection but does not delete the collection's data file(s). You can format and create reports from the data file(s) as if the data collection had run to completion. DECtrace also writes a message to the history database indicating that the data collection has been cancelled. You can examine the history database at any time with the SHOW HISTORY command. See Section 3.5.2 for information about the history database and the SHOW HISTORY command.

When you cancel a collection it does not terminate immediately, but is set to an aborting state. For an active collection, the Registrar process (or processes in a VAXcluster environment) sends a "Stop collecting" message to each process that is recording event data. This communication takes place very quickly, but if you cancel a collection and immediately enter the SHOW COLLECTION command, your collection will still exist in an aborting state. A collection that is aborting is indicated in the SHOW COLLECTION display by two asterisks (**) next to the entry. Whenever you cancel an active collection you receive an informational message from the DECtrace Registrar stating that the collection has been set to the aborting state:

```
$  COLLECT CANCEL COLLECTION TEMP /NOCONFIRM
%EPC-S-SCHED_ABTNG, Data collection TEMP has been set to aborting
```

In a cluster, CANCEL COLLECTION cancels data collection either locally or cluster-wide depending on how the collection was originally scheduled. If you scheduled data collection with the /NOCLUSTER qualifier, CANCEL COLLECTION cancels the collection at the local node. If you scheduled data collection with the /CLUSTER qualifier, CANCEL COLLECTION cancels the collection on all nodes in the cluster. Note that you cannot cancel collections from individual nodes in a cluster environment if you scheduled the collection using /CLUSTER.

The following example cancels the collection MYTEST, which was originally schedule to run on the local node:

```
$  COLLECT CANCEL COLLECTION MY_TEST /NOCONFIRM
%EPC_S_SCHED_CANCEL, Collection MY_TEST is cancelled.
```

You can cancel all of the collections (active or pending) that use a particular facility selection if you specify the /SELECTION qualifier. This option is useful if you typically schedule multiple data collections referencing the same facility selection. It is also needed when you want to delete a facility selection. To do this, you must first cancel all collections using that selection.

The following example cancels all data collections that are scheduled with the facility selection TEMP_SELECTION:

```
$ COLLECT CANCEL COLLECTION /SELECTION=TEMP_SELECTION /NOCONFIRM
%EPC_S_SCHED_CANCEL, Collection MY_COLL_MYVAX1 is cancelled.
%EPC_S_SCHED_CANCEL, Collection MY_COLL_MYVAX2 is cancelled.
%EPC_S_SCHED_CANCEL, Collection TEMP_TEST is cancelled.
```

3.5 Displaying Information about Data Collection

This section describes the commands that allow you to examine the status of data collection on your system. The actual recording of event data is the third step (see Figure 1–1) in the DECtrace collection process. You can examine the schedule of active and pending collections, and you can display any errors, warnings, or informational messages encountered by the collections.

3.5.1 Displaying the Schedule for Data Collection

Collections exist on your system in either an active or a pending state. Because only one collection can be active on a particular node at any time, it is important to know when data collection is scheduled to occur.

The DECtrace administration database maintains the schedule of data collection on your VAXcluster. You can display information about scheduled collections with the SHOW COLLECTION command.

Example 3–5 shows a sample of the display produced with the SHOW COLLECTION/FORMAT=BRIEF command. The arrow (->) next to the line describing the MY_FULL_TEST collection indicates that the collection is active. Note that only one collection can be active on a node at any time.

Example 3–5 Display for SHOW COLLECTION Using the Brief Format

```
9-MAY-1989 10:09            Scheduled Collections                   Page 1
Brief Report                                                  DECtrace V1.0

Collections scheduled for the entire cluster

    Selection Name   Collection Name  Start            End
    ---------------  ---------------  ---------------  ---------------
->  MY_SELECTION     MY_FULL_TEST      9-MAY-89 10:00   9-MAY-89 11:00

DECtrace Collection Schedule for node MYVAX1

    Selection Name   Collection Name  Start            End
    ---------------  ---------------  ---------------  ---------------
    MY_SELECTION     MY_COLL_MYVAX1    9-MAY-89 12:00   9-MAY-89 13:00

DECtrace Collection Schedule for node MYVAX2

    Selection Name   Collection Name  Start            End
    ---------------  ---------------  ---------------  ---------------
    MY_SELECTION     MY_COLL_MYVAX2    9-MAY-89 12:00   9-MAY-89 13:00

DECtrace Collection Schedule for node SMTHVX

    Selection Name   Collection Name  Start            End
    ---------------  ---------------  ---------------  ---------------
    MY_SELECTION     MY_LOCAL_TEST     9-MAY-89 11:00   9-MAY-89 12:00
```

3.5.2 Displaying the History for Collections

The DECtrace **history database** contains a record of the informational and error messages that are encountered during data collection. The SHOW HISTORY command allows you to discover if any errors occurred during a collection (or many collections). You can also display any informational messages that occurred during data collection. An example of an informational message is a confirmation message sent back to DECtrace by a process that has actually begun data collection, or a message that a process has registered with DECtrace and is available for collection. Displaying informational messages is most useful for facility developers and support personnel, while displaying errors is useful for general users.

The format of the SHOW HISTORY command is:

SHOW HISTORY collection-name
[*/BEFORE="time"*
/[NO]CLUSTER
/FORMAT=type
/NODE=node-name
/OUTPUT=file-spec
/SINCE="time"]

Valid format types are ERROR, INFORMATIONAL, and ALL which includes both errors and informational messages.

You should examine the history database during and after data collection to verify that the collection started successfully and did not encounter any serious errors. You should always examine the history before formatting your collected data. Note that you can format data files from collections that failed during active data collection, but you might not achieve the full results that you wanted.

Note that an error encountered by a process which has registered with DECtrace is not related to any collection for which the process might be recording event data. If you use the COLLECTION-NAME parameter to the SHOW HISTORY command, you do not see errors or messages generated by the individual processes. For example, if a process is unable to record data due to a file protection violation on the data collection file, the message is on the SHOW HISTORY /FORMAT=ALL report; not the report for a specific collection.

Example 3–6 shows the display for the SHOW HISTORY/FORMAT=ERROR command and Example 3–7 shows the display for the SHOW HISTORY /FORMAT=INFORMATIONAL command.

Example 3–6 Display for SHOW HISTORY /FORMAT=ERROR

```
04-APR-1990 10:03           Data Collection History                 Page 1
DECtrace error history for cluster                         DECtrace V1.0

Collection:

Date and Time            EPID      Process Name     Registration Id
-----------------------  --------  ---------------  --------------------
03-APR-1990 12:34:15.41  31600062  DBMS_USER1
   %SYSTEM-W-DEVICEFULL, device full - allocation failure
   %EPC-E-HST_PRCERR, Error received from collecting process

03-APR-1990 12:34:16.48  31600062  ACMS_USER1       Personnel
   %SYSTEM-W-DEVICEFULL, device full - allocation failure
   %EPC-E-HST_PRCERR, Error received from collecting process

04-APR-1990 09:47:42.73  316000D6  DB_MANAGER
   %EPC-E-OPEDCF, Error opening data collection file
   %EPC-E-HST_PRCERR, Error received from collecting process

04-APR-1990 09:47:47.07  316000D6  DB_MANAGER
   %EPC-E-HST_PRCERR, Error received from collecting process
   %SYSTEM-F-NOPRIV, no privilege for attempted operation
```

Example 3–7 Display for SHOW HISTORY /FORMAT=INFORMATIONAL

```
04-APR-1990 10:05            Data Collection History                Page 1
DECtrace informational history for cluster                    DECtrace V1.0

Collection:  EPC$IVP_COLLECTION
Node:  MYVAX1

Date and Time            EPID      Process Name     Registration Id
-----------------------  --------  ---------------  --------------------
03-APR-1990 15:03:24.74  38A00B79  _RTA16:
   %EPC-S-HST_SCHED, Data collection scheduled

03-APR-1990 15:03:29.38  38A0124E  EPC$REGISTRAR    EPC$IVP_SELECTION
   %EPC-S-HST_START, Collecting started

03-APR-1990 15:03:53.55  38A00B79  _RTA16:
   %EPC-S-HST_START_COLL, Process started collecting data

03-APR-1990 15:04:55.20  38A00B79  _RTA16:
   %EPC-S-HST_ABORT, Collection aborting

03-APR-1990 15:04:58.04  38A0124E  EPC$REGISTRAR
   %EPC-S-HST_END, Collecting ended

03-APR-1990 15:05:01.50  38A0124E  EPC$REGISTRAR
   %EPC-S-HST_DELETED, Collection deleted
```

4 Generating Reports

The DECtrace software can generate statistical reports based on your collected data. This chapter describes how to manipulate your raw data collection files into formatted reports. It describes how to:

- Merge data collection files
- Format data collection files into either an Rdb/VMS database or a VAX RMS file
- Generate reports based on the formatted Rdb/VMS database

Formatting and reporting are the fourth and fifth steps (see Figure 1–1) for general users of DECtrace. The final step is interpretation and analysis of the DECtrace reports.

Table 4–1 summarizes the DECtrace commands available to accomplish the functions associated with reporting.

Table 4–1 Commands for Generating Reports

Command	Description
FORMAT	Formats one or more DECtrace data files into a formatted data file or database.
REPORT	Generates a report based on formatted data from one or more collections.

4.1 Overview of Reporting

After data collection is performed, the data collection files can be merged and formatted into a single Rdb/VMS database or VAX RMS file. The formatted database or file can contain data from one or more different collections as long as an identical facility selection was used for each collection. The collections can have run on the same node or on many different nodes. This capability provides a convenient way to maintain all the data collected for one or several collections.

You can write your own customized reports using the published formatted database and file layouts described in Appendix A.

4.2 Merging and Formatting Your Data Files

The FORMAT command is used to merge and format collected data. The first step in formatting is to decide whether you want to generate an Rdb/VMS database or a VAX RMS file. The advantage of generating an Rdb/VMS database is that you can generate DECtrace reports. In either case, you can generate reports using a fourth-generation language or programming language containing embedded calls to either Rdb/VMS or VAX RMS.

If you choose to format the data into an Rdb/VMS database, you can optionally choose to store the record definitions in VAX CDD/Plus using the /CDDPLUS_DEFINITIONS qualifier. Of course, this feature is only available if CDD/Plus is currently installed on your system. If you format the data collection files without the /CDDPLUS_DEFINITIONS qualifier and later decide that you want to use CDD/Plus, you can use the INTEGRATE command of RDO or SQL.

4.2.1 Merging and Formatting the Collected Data into an Rdb/VMS Database

Multiple data collection files can be merged into a single formatted database. The following example formats the data in two files (COLL1_FILE1 and COLL1_FILE2) into an Rdb/VMS database called COLL_DB.

```
$ COLLECT FORMAT/TYPE=RDBVMS COLL1_FILE1,COLL1_FILE2 USER1:COLL_DB
```

If you wish to merge two more data collection files from another collection (using the same facility selection) into the same Rdb/VMS database that was just created, issue the following command:

```
$ COLLECT FORMAT/MERGE/TYPE=RDBVMS COLL2_FILE1,COLL2_FILE2 -
_$ USER1:COLL_DB
```

Formatting of a single data collection file is done using batch update transaction mode. This speeds up the formatting operation, especially for very large files. When formatting several data collection files into a single formatted database, you should format the largest file first and then merge the remaining files into the new database.

4.2.2 Merging and Formatting the Collected Data into a VAX RMS File

To merge two data collection files from a single collection into a new VAX RMS file called COLL_FILE, issue the following command:

```
$ COLLECT FORMAT/TYPE=RMS COLL1_FILE1,COLL1_FILE2 USER1:COLL_FILE
```

If you want to merge two more data collection files from another collection (using the same facility selection) into the same VAX RMS file that was just created, issue the following command:

```
$ COLLECT FORMAT/MERGE/TYPE=RMS COLL2_FILE1,COLL2_FILE2 -
_$ USER1:COLL_FILE
```

4.3 Formatting Optimization

You can improve response time for formatting data collection files into an Rdb /VMS database by specifying a set of optimization parameters on the FORMAT command. The parameters are part of the /RDBVMS_OPTIMIZATION qualifier, which has the following format:

/RDBVMS_OPTIMIZATION = (parameter=value[, . . .])

Table 4–2 shows the optimization parameters and their defaults.

Table 4–2 Format Optimization Parameters

Optimization Parameter	Default Value
ALLOCATION	2000 database pages
BUFFER_SIZE	30 blocks
[NO]JOURNAL	JOURNAL
MIN_EXTENT	500 pages
NUM_BUFFERS	30 buffers
[NO]STRING_OPTIMIZATION	STRING_OPTIMIZATION
[NO]VIEWS	VIEWS

The following example uses the optimization parameter to format a large data collection file into a formatted Rdb/VMS database:

```
$ COLLECT FORMAT VERY_BIG_FILE.DAT BIG_DATABASE -
_$ /RDBVMS_OPTIMIZATION=(NOJOURNAL, ALLOCATION=10000 -
_$ STRING_OPTIMIZATION=(FIRST=32, SEGMENT=64) -
_$ NOVIEWS)
```

See the description of the FORMAT command in Chapter 7 for a description of each optimization parameter.

4.4 Report Types

DECtrace can generate tabular reports based on the data in an Rdb/VMS formatted database. Table 4–3 lists the three different types of reports that you can produce.

Table 4–3 DECtrace Reports

Type of Report	Description
DETAIL	Actual values of the items collected for each event.
FREQUENCY	Event occurrence summary based on a selected time interval.
SUMMARY (default)	Summary statistics about the collected data.

The example reports in this section use COBOL-like pictures to describe the format of the data in the reports. Table 4–4 shows what each data format means.

Table 4–4 Data Formats

Format	Meaning
DD-MMM-YYYY	Date format (for example, 09-May-1989).
HH:MM:SS.HH	Time format (for example, 11:42:23.98).
X	A printable ASCII character within a text field.
9	A numeric character, 0 through 9[1].
Z	A numeric character, 0 through 9[1]. Leading zeros are replaced with blanks. If the numeric field is negative, a minus sign is displayed immediately to the left of the most significant digit.
. (period)	A decimal point.

[1]If a numeric value overflows the field, then the field is filled with asterisk (*) characters.

In the examples of reports, an exclamation point (!) indicates a comment about the layout of the report. The comment describes the line immediately to its left. Note that the comment does not appear in the actual report.

4.4.1 Detail Report

The Detail Report is useful for application developers (for debugging purposes) who want to see the actual values of the items for each start, end, or point event record. The display for each event on the Detail Report consists of the time and date stamp when the event record was recorded, and all of the report items for the event records meeting the selection criteria specified on the REPORT command. If the event was a duration event, the report displays the values of the items collected on both the start and end events.

Example 4–1 shows the format of a basic report. Two events are displayed in this report, each with three items (one string and two numeric items). The first event is a point event for which five occurrences were recorded. The second event is a duration event for which the items associated with both the start and end events for one event occurrence are displayed. An index page, which follows all reports consisting of more than one event, is not shown in the example.

Example 4–1 Display for Detail Report

```
DD-MMM-YYYY HH:MM XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  PageZZZ9
Selection: XXXXXXXXXXXXXXX                                        DECtrace V1.0-0

Event XXXXXXXXXXXXXX  In Facility XXXXXXXX  Version XXXXXXXXX ! Event 1
  For Collections XXXX,XXXX
  For Nodes XXXXXX,XXXXXX               !
  For PID   XXXXXX,XXXXXX               ! These lines indicate optional
  For Image XXXXXX,XXXXXX               ! restrictions placed on the report
  With XXXXXXXXXXXXX = XXXXXXXXXX       ! using the RESTRICTIONS option
  And  XXXXXXXXXX = XXXXXXX             !

! For Point event.
Timestamp                XXXXXXXXXXXX  XXXXXXXXX  XXXXXXXXX  ! Item Headers

DD-MMM-YYYY HH:MM:SS.HH  XXXXXXXXXXXX  ZZZZZZZZ9  ZZZZZZZZ9  ! 1st occurrence
DD-MMM-YYYY HH:MM:SS.HH  XXXXXXXXXXXX  ZZZZZZZZ9  ZZZZZZZZ9  ! 2nd occurrence
DD-MMM-YYYY HH:MM:SS.HH  XXXXXXXXXXXX  ZZZZZZZZ9  ZZZZZZZZ9    .
DD-MMM-YYYY HH:MM:SS.HH  XXXXXXXXXXXX  ZZZZZZZZ9  ZZZZZZZZ9    .
DD-MMM-YYYY HH:MM:SS.HH  XXXXXXXXXXXX  ZZZZZZZZ9  ZZZZZZZZ9    .

DD-MMM-YYYY HH:MM XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  PageZZZ9
Selection: XXXXXXXXXXXXXXX                                        DECtrace V1.0-0

Event XXXXXXXXXXXXXX  In Facility XXXXXXXX  Version XXXXXXXXX ! Event 2
  For Collections XXXX,XXXX
  For Nodes XXXXXX,XXXXXX
  With XXXXXXXXXXXXX = XXXXXXXXXX
  And  XXXXXXXXXX = XXXXXXX
```

(continued on next page)

Example 4–1 (Cont.) Display for Detail Report

```
! For Duration Event

Timestamp                XXXXXXXXXXXX  XXXXXXXXX  XXXXXXXXX  ! Item Headers

DD-MMM-YYYY HH:MM:SS.HH  XXXXXXXXXXXX  ZZZZZZZZ9  ZZZZZZZZ9  ! Start event
DD-MMM-YYYY HH:MM:SS.HH  XXXXXXXXXXXX  ZZZZZZZZ9  ZZZZZZZZ9  ! End event
```

Headers and values for text are left justified. The headers and values for numeric report items are right justified. See Section 4.5.4 for an explanation of how this text is wrapped.

The item headers are defined in the facility definition. You can override the default header text and width using the reporting options described in Section 4.8.

Example 4–2 shows a Detail Report based on data collected from the ATM sample application. The following example shows the command used to generate the report:

```
$ COLLECT REPORT ATM_DATA.RDB /TYPE=DETAIL
```

Example 4–2 Sample Detail Report Based on ATM Data

```
16-FEB-1990 16:46                Detail Report                     Page 1
Selection: JUST_ATM                                          DECtrace V1.0-0

Event:  BALANCE_EVENT   In Facility:  ATM_SAMPLE     Version:  V1.0

Timestamp                    Elapsed  BUFFERED IO     CPU TIME  CURREN
                                                                T PRIO

16-FEB-1990 16:16:11.87        3.42         3966         1509        9

16-FEB-1990 16:16:15.29                     3979         1511        9

  DIRECT IO   PAGEFAULTS     PAGEFAULT      VIRTUAL    GLOBAL WS   PRIVATE WS
                                   IOs         SIZE

        381         5679           129         5112           92          397

        381         5686           129         5112           99          397

    WORKING
    SET SIZ

       1109

       1109
```

(continued on next page)

Example 4–2 (Cont.) Sample Detail Report Based on ATM Data

```
                =================  Next Occurrence  ====================

Timestamp                  Elapsed  BUFFERED IO     CPU TIME   CURREN
                                                               T PRIO

16-FEB-1990 16:16:23.16       0.96         4033         1517        9

16-FEB-1990 16:16:24.12                    4046         1519        9

  DIRECT IO   PAGEFAULTS     PAGEFAULT      VIRTUAL    GLOBAL WS    PRIVATE WS
                                   IOs         SIZE

        381         5695           129         5112          108           397

        381         5695           129         5112          108           397

    WORKING
    SET SIZ

       1109

       1109

                =================  Next Occurrence  ====================
                                        .
                                        .
                                        .


16-FEB-1990 16:46                Detail Report                            Page 6
Selection: JUST_ATM                                               DECtrace V1.0-0

Event:  DEPOSIT_EVENT    In Facility:  ATM_SAMPLE      Version:  V1.0

Timestamp                  Elapsed  BUFFERED IO     CPU TIME   CURREN
                                                               T PRIO

16-FEB-1990 16:16:18.67       3.38         3998         1513        9

16-FEB-1990 16:16:22.05                    4014         1515        9

  DIRECT IO   PAGEFAULTS     PAGEFAULT      VIRTUAL    GLOBAL WS    PRIVATE WS
                                   IOs         SIZE

        381         5686           129         5112           99           397

        381         5695           129         5112          108           397

    WORKING
    SET SIZ

       1109

       1109
```

(continued on next page)

Example 4–2 (Cont.) Sample Detail Report Based on ATM Data

```
                ================ Next Occurrence ===================

Timestamp                    Elapsed  BUFFERED IO     CPU TIME  CURREN
                                                                T PRIO

16-FEB-1990 16:17:03.31        6.08          4256        1544        9

16-FEB-1990 16:17:09.39                      4272        1546        9

  DIRECT IO   PAGEFAULTS    PAGEFAULT      VIRTUAL    GLOBAL WS   PRIVATE WS
                                  IOs         SIZE

        381         5695          129         5112          108          397

        381         5695          129         5112          108          397

    WORKING
    SET SIZ

       1109

       1109

                ================ Next Occurrence ===================
                                      .
                                      .
                                      .


16-FEB-1990 16:46                Detail Report                          Page 8
Selection: JUST_ATM                                            DECtrace V1.0-0

Event:  WITHDRAW_EVENT  In Facility:  ATM_SAMPLE      Version:  V1.0

Timestamp                    Elapsed  BUFFERED IO     CPU TIME  CURREN
                                                                T PRIO

16-FEB-1990 16:16:46.19        2.34          4127        1531        9

16-FEB-1990 16:16:48.53                      4143        1532        9

  DIRECT IO   PAGEFAULTS    PAGEFAULT      VIRTUAL    GLOBAL WS   PRIVATE WS
                                  IOs         SIZE

        381         5695          129         5112          108          397

        381         5695          129         5112          108          397

    WORKING
    SET SIZ

       1109

       1109
```

(continued on next page)

Example 4–2 (Cont.) Sample Detail Report Based on ATM Data

```
16-FEB-1990 16:46                 Detail Report                      Page 9
Selection: JUST_ATM                                          DECtrace V1.0-0

Report Index

Facility Name                  Event Name                      Page

ATM_SAMPLE                     BALANCE_EVENT                   1
ATM_SAMPLE                     DEPOSIT_EVENT                   6
ATM_SAMPLE                     WITHDRAW_EVENT                  8
```

4.4.2 Frequency Report

The Frequency Report presents the number of event occurrences for each event, which match the selection criteria specified on the REPORT command. The layout of the report is similar to the Summary Report (see Section 4.4.3) except that it displays time period occurrences instead of statistics.

Generally, you use the /INTERVAL qualifier when you generate Frequency Reports. This qualifier specifies the time interval for which event occurrences are calculated. The time interval must be one second, one minute, or one hour, specified as one of the following keywords:

SECONDS
MINUTES (default)
HOURS

Example 4–3 shows the format of a basic report generated using the /INTERVAL=SECONDS qualifier. A count is displayed for each second during which at least one event occurrence was recorded. An index page, which follows all reports consisting of more than one event, is not shown in the example.

Example 4–3 Display for Frequency Report Using /INTERVAL=SECONDS

```
DD-MMM-YYYY HH:MM XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  PageZZZ9
Selection: XXXXXXXXXXXXXXX                                   DECtrace V1.0-0

Event XXXXXXXXXXXXXX  In Facility XXXXXXXX  Version XXXXXXXXX  ! Event 1
  For Collections XXXX,XXXX
  For Nodes XXXXXX,XXXXXX              !
  For PID   XXXXXX,XXXXXX              ! These lines indicate optional
  For Image XXXXXX,XXXXXX              ! restrictions placed on the report
  With XXXXXXXXXXXXX = XXXXXXXXXX      ! using the RESTRICTIONS option
  And  XXXXXXXXXX = XXXXXXX            !

Time Period           Occurrences

DD-MMM-YYYY HH:MM:SS  ZZZZZZZZZZ9
DD-MMM-YYYY HH:MM:SS  ZZZZZZZZZZ9
DD-MMM-YYYY HH:MM:SS  ZZZZZZZZZZ9
DD-MMM-YYYY HH:MM:SS  ZZZZZZZZZZ9


DD-MMM-YYYY HH:MM XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  PageZZZ9
Selection: XXXXXXXXXXXXXXX                                   DECtrace V1.0-0

Event XXXXXXXXXXXXXX  In Facility XXXXXXXX  Version XXXXXXXXX ! Event 2
  For Collections XXXX,XXXX
  For Nodes XXXXXX,XXXXXX
  For PID   XXXXXX,XXXXXX
  For Image XXXXXX,XXXXXX
  With XXXXXXXXXXXXX = XXXXXXXXXX
  And  XXXXXXXXXX = XXXXXXX

Time Period           Occurrences

DD-MMM-YYYY HH:MM:SS  ZZZZZZZZZZ9
DD-MMM-YYYY HH:MM:SS  ZZZZZZZZZZ9
DD-MMM-YYYY HH:MM:SS  ZZZZZZZZZZ9
DD-MMM-YYYY HH:MM:SS  ZZZZZZZZZZ9
DD-MMM-YYYY HH:MM:SS  ZZZZZZZZZZ9
DD-MMM-YYYY HH:MM:SS  ZZZZZZZZZZ9
```

For duration events, only occurrences of completed start and end pairs are counted. A diagnostic message is presented for incomplete pairs.

Example 4–4 shows a Frequency Report based on data collected from the ATM sample application. The following example shows the command used to generate the report:

```
$ COLLECT REPORT ATM_DATA.RDB /TYPE=FREQUENCY -
_$ /INTERVAL=MINUTES
```

Although this report provides the least information of the three report types, it is useful for tracing the execution of an instrumented application. For example, in the ATM report it is apparent that the BALANCE_EVENT occurs much more frequently than the other events. By examining the time-stamps on the events (perhaps generating a new report using /INTERVAL=SECONDS), it becomes evident that customers are checking their account balance before and after each transaction. Based on this information, the application programmer might choose to modify the ATM program to automatically display the new balance after each transaction.

Example 4–4 Sample Frequency Report Based on ATM Data

```
16-FEB-1990 16:43                 Frequency Report                       Page 1
Selection: JUST_ATM                                               DECtrace V1.0-0

Event:  BALANCE_EVENT   In Facility:  ATM_SAMPLE      Version:  V1.0

Time Period          Occurrences

16-FEB-1990 16:16:00          5
16-FEB-1990 16:17:00          1
16-FEB-1990 16:20:00          4


16-FEB-1990 16:43                 Frequency Report                       Page 2
Selection: JUST_ATM                                               DECtrace V1.0-0

Event:  DEPOSIT_EVENT   In Facility:  ATM_SAMPLE      Version:  V1.0

Time Period          Occurrences

16-FEB-1990 16:16:00          1
16-FEB-1990 16:17:00          1

%EPC-I-NOEND, 1 Start Event Records had no matching End


16-FEB-1990 16:43                 Frequency Report                       Page 3
Selection: JUST_ATM                                               DECtrace V1.0-0

Event:  WITHDRAW_EVENT  In Facility:  ATM_SAMPLE      Version:  V1.0

Time Period          Occurrences

16-FEB-1990 16:16:00          1


16-FEB-1990 16:43                 Frequency Report                       Page 4
Selection: JUST_ATM                                               DECtrace V1.0-0

Report Index

Facility Name                 Event Name                        Page

ATM_SAMPLE                    BALANCE_EVENT                     1
ATM_SAMPLE                    DEPOSIT_EVENT                     2
ATM_SAMPLE                    WITHDRAW_EVENT                    3
```

4.4.3 Summary Report

The Summary Report presents up to 7 different statistics for each report item that is specified in the selection criteria of the REPORT command. The statistics are: maximum, minimum, mean, standard deviation[1], count (number of occurrences), total, and 95th percentile[2]. The sample standard deviation is computed by the following formula, where n is the sample size and X_i is the value of each sample element:

$$Sample_i = \sqrt{\frac{n\sum_{i=1}^{n} X_i^2 - (\sum_{i=1}^{n} X_i)^2}{n(n-1)}}$$

The Summary Report contains an event group for each value of a groupable item and for all events with the same event name. This capability allows for subtotal summaries to be calculated for each groupable item. This also allows grand total summaries to be calculated for each event name. Because Example 4–5 contains 2 groupable items, there is a separate event group for each different value of the second groupable item. This is followed by an event group which presents a subtotal of the report items for all event groups for each value of the first groupable item. At the end of all summary groups for an event name, a summary group presents a grand total for all report items.

Table 4–5 describes the parts contained in each event group on the Summary Report.

[1] The various subvalues used to calculate the sample standard deviation are stored as D-floating data types. They have a range of 0.29 * 10E-38 to 0.29 * 10E38 with 16 decimal places of precision. The major implication is that if the squared elements summed requires more than 16 decimal places, a roundoff error may occur.

[2] The 95th percentile is based on the standard deviation. It assumes a relatively normal distribution with a sample size, n, greater than 1000. Based on sample sizes greater than 1000, a confidence level of 95% can be computed as 1.96 units away from the standard deviation.

Table 4–5 Parts of a Summary Report Event Group

Part	Meaning
Groupable item values	Each value is on a separate line and left justified. The headers that precede each groupable item are aligned. See Section 4.5.1 for an explanation for how these items are wrapped. The groupable item values are followed by a blank line.
Report item headers	The text of the header is defined in the facility definition for the event or can be specified by the user (see the ITEM report option for details). The headers for the text items are left justified. Numeric headers are right justified. See Section 4.5.3 for an explanation of how headers are wrapped. Each report item is separated by two blank columns.
Statistics	The format for the Statistics are:

Statistic	Elapsed Time	Others
Minimum	ZZZZZ9.99	ZZZZZZZZ9
Maximum	ZZZZZ9.99	ZZZZZZZZ9
Mean	ZZZZZ9.99	ZZZZZ9.99
Std Dev[1]	ZZZZZ9.99	ZZZZZ9.99
95 Prct[2]	ZZZZZ9.99	ZZZZZ9.99
Total	ZZZZZZZZ9	ZZZZZZZZ9
Count[3]	ZZZZZZZZ9	

The Statistics are followed by a blank line.

[1]Std Dev is Standard Deviation.

[2]95 Prct is 95th Percentile.

[3]Because the Count is the same for all report items, it is only displayed for the first report item.

If any of the event records for duration events have unmatched start and end event pairs, a diagnostic message displays following the event. This message indicates how many unmatched event records are encountered. Note that the total of all unmatched event records for the entire report is included with the grand totals.

Example 4–5 shows the structure of a Summary Report for two different types of events. In each case, two groupable items and five report items are specified in the REPORT options. The first groupable item is a 12-character text string. The second groupable item is a longword data type. The five report items are longword data types. All the statistics are requested for the first event. Only the minimum, maximum, mean, and count statistics are requested for the last event. The next to the last event group has incomplete start and end event record pairs; hence, the two diagnostic messages. An index page, which follows all reports consisting of more than one event, is not shown in the example.

Example 4–5 Display for Summary Report

```
DD-MMM-YYYY HH:MM XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  PageZZZ9
Selection: XXXXXXXXXXXXXXX                                     DECtrace V1.0-0

Event XXXXXXXXXXXXXX  In Facility XXXXXXXX  Version XXXXXXXXX ! First event
  For Collections XXXX,XXXX
  For Nodes XXXXXX,XXXXXX                   !
  For PID   XXXXXX,XXXXXX                   ! These lines indicate optional
  For Image XXXXXX,XXXXXX                   ! restrictions placed on the report
  With XXXXXXXXXXXXX = XXXXXXXXXX           ! using the RESTRICTIONS option
  And  XXXXXXXXXX = XXXXXXX                 !

    XXXXXX XXXX: XXXXXXXXXXXX               ! Value of first Groupable Item
XXXXXXXXX XXXXX: ZZZZZZZZ9                  ! Value of second Groupable Item

         XXXXXXXXX  XXXXXXXXX  XXXXXXXXX  XXXXXXXXX  XXXXXXXXX  ! Item Headers

Minimum  ZZZZZ9.99  ZZZZZ9.99  ZZZZZZZZ9  ZZZZZZZZ9  ZZZZZZZZ9
Maximum  ZZZZZ9.99  ZZZZZ9.99  ZZZZZZZZ9  ZZZZZZZZ9  ZZZZZZZZ9
Mean     ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99
Std Dev  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99
95 Prct  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99
Total    ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99
Count    ZZZZZZZZ9

    XXXXXX XXXX: XXXXXXXXXXXX               ! Value of first Groupable Item
XXXXXXXXX XXXXX: ZZZZZZZZ9                  ! Value of second Groupable Item

         XXXXXXXXX  XXXXXXXXX  XXXXXXXXX  XXXXXXXXX  XXXXXXXXX  ! Item Headers

Minimum  ZZZZZ9.99  ZZZZZ9.99  ZZZZZZZZ9  ZZZZZZZZ9  ZZZZZZZZ9
Maximum  ZZZZZ9.99  ZZZZZ9.99  ZZZZZZZZ9  ZZZZZZZZ9  ZZZZZZZZ9
Mean     ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99
Std Dev  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99
95 Prct  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99
Total    ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99
Count    ZZZZZZZZ9

=============
Subtotals For:
=============
  XXXXXX XXXX: XXXXXXXXXXXX ! This group is a subtotal summary
                            ! of the first groupable item
         XXXXXXXXX  XXXXXXXXX  XXXXXXXXX  XXXXXXXXX  XXXXXXXXX  ! Item Headers

Minimum  ZZZZZ9.99  ZZZZZ9.99  ZZZZZZZZ9  ZZZZZZZZ9  ZZZZZZZZ9
Maximum  ZZZZZ9.99  ZZZZZ9.99  ZZZZZZZZ9  ZZZZZZZZ9  ZZZZZZZZ9
Mean     ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99
Std Dev  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99
95 Prct  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99
Total    ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99
Count    ZZZZZZZZ9
```

(continued on next page)

Example 4–5 (Cont.) Display for Summary Report

```
               ================ Grand Total ===================
         XXXXXXXXX  XXXXXXXXX  XXXXXXXXX  XXXXXXXXX  XXXXXXXXX  ! Item Headers

Minimum  ZZZZZ9.99  ZZZZZ9.99  ZZZZZZZZ9  ZZZZZZZZ9  ZZZZZZZZ9
Maximum  ZZZZZ9.99  ZZZZZ9.99  ZZZZZZZZ9  ZZZZZZZZ9  ZZZZZZZZ9
Mean     ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99
Std Dev  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99
95 Prct  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99
Total    ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99
Count    ZZZZZZZZ9  ! This group summarizes the first event

DD-MMM-YYYY HH:MM XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  PageZZZ9
Selection: XXXXXXXXXXXXXXX                                     DECtrace V1.0-0

Event XXXXXXXXXXXXXX  In Facility XXXXXXXX  Version XXXXXXXXX  ! Second event
  In Collection Runs XXXX,XXXX
  For Nodes XXXXXX,XXXXXX
  With XXXXXXXXXXXXX = XXXXXXXXXX
  And  XXXXXXXXXX = XXXXXXX

    XXXXXX XXXX: XXXXXXXXXXXX         ! Value of first Groupable Item
XXXXXXXXX XXXXX: ZZZZZZZZ9            ! Value of second Groupable Item

         XXXXXXXXX  XXXXXXXXX  XXXXXXXXX  XXXXXXXXX  XXXXXXXXX  ! Item Headers

Minimum  ZZZZZ9.99  ZZZZZ9.99  ZZZZZZZZ9  ZZZZZZZZ9  ZZZZZZZZ9
Maximum  ZZZZZ9.99  ZZZZZ9.99  ZZZZZZZZ9  ZZZZZZZZ9  ZZZZZZZZ9
Mean     ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99
Total    ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99
Count    ZZZZZZZZ9

    XXXXXX XXXX: XXXXXXXXXXXX ! This group is a subtotal summary of the first
                              ! groupable item
         XXXXXXXXX  XXXXXXXXX  XXXXXXXXX  XXXXXXXXX  XXXXXXXXX  ! Item Headers

Minimum  ZZZZZ9.99  ZZZZZ9.99  ZZZZZZZZ9  ZZZZZZZZ9  ZZZZZZZZ9
Maximum  ZZZZZ9.99  ZZZZZ9.99  ZZZZZZZZ9  ZZZZZZZZ9  ZZZZZZZZ9
Mean     ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99
Total    ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99
Count    ZZZZZZZZ9

               ================ Grand Total ===================
         XXXXXXXXX  XXXXXXXXX  XXXXXXXXX  XXXXXXXXX  XXXXXXXXX  ! Item Headers

Minimum  ZZZZZ9.99  ZZZZZ9.99  ZZZZZZZZ9  ZZZZZZZZ9  ZZZZZZZZ9
Maximum  ZZZZZ9.99  ZZZZZ9.99  ZZZZZZZZ9  ZZZZZZZZ9  ZZZZZZZZ9
Mean     ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99
Total    ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99  ZZZZZ9.99
Count    ZZZZZZZZ9  ! This group summarizes the second event

%EPC-I-NOSTART, ZZZZZZZZ9 End Event Records had no matching Start
%EPC-I-NOEND, ZZZZZZZZ9 Start Event Records had no matching End
```

Example 4–6 shows a Summary Report based on data collected from the ATM sample application. The following example shows the command used to generate the report:

```
$ COLLECT REPORT ATM_DATA.RDB /TYPE=SUMMARY -
_$ /STATISTICS=ALL
```

Example 4–6 Sample Summary Report Based on ATM Data

```
30-APR-1990 14:19                Summary Report                     Page 1
Selection: JUST_ATM                                          DECtrace V1.0-0

Event:  BALANCE_EVENT   In Facility:  ATM_SAMPLE      Version:  V1.0

           Elapsed  BUFFERED IO     CPU TIME  CURREN     DIRECT IO   PAGEFAULTS
                                              T PRIO

Minimum       0.64           13            0       9            0            0
Maximum       4.79           13            3       9            0            8
Mean          1.78        13.00         2.10    9.00         0.00         1.50
Std Dev       1.39         0.00         0.87    0.00         0.00         3.17
95 Prct       4.52        13.00         3.81    9.00         0.00         7.71
Total        17.84          130           21      90            0           15
Count           10

           PAGEFAULT       VIRTUAL    GLOBAL WS    PRIVATE WS       WORKING
                 IOs          SIZE                                  SET SIZ

Minimum            0          5112           99           380          1109
Maximum            0          5112          108           397          1109
Mean            0.00       5112.00       106.70        390.20       1109.00
Std Dev         0.00          0.00         2.75          8.77          0.00
95 Prct         0.00       5112.00       112.09        407.40       1109.00
Total              0         51120         1067          3902         11090
Count             10

%EPC-I-RPQU_BAD_95, 95 Prct for events with counts under 1000 are less precise


30-APR-1990 14:19                Summary Report                     Page 2
Selection: JUST_ATM                                          DECtrace V1.0-0

Event:  DEPOSIT_EVENT   In Facility:  ATM_SAMPLE      Version:  V1.0

           Elapsed  BUFFERED IO     CPU TIME  CURREN     DIRECT IO   PAGEFAULTS
                                              T PRIO

Minimum       3.38           16            2       9            0            0
Maximum       6.08           16            2       9            0            9
Mean          4.73        16.00         2.00    9.00         0.00         4.50
Std Dev       1.90         0.00         0.00    0.00         0.00         6.36
95 Prct       8.47        16.00         2.00    9.00         0.00        16.97
Total         9.46           32            4      18            0            9
Count            2
```

(continued on next page)

Example 4–6 (Cont.) Sample Summary Report Based on ATM Data

```
          PAGEFAULT      VIRTUAL    GLOBAL WS   PRIVATE WS      WORKING
                IOs         SIZE                                SET SIZ

Minimum           0         5112         108          397          1109
Maximum           0         5112         108          397          1109
Mean           0.00      5112.00      108.00       397.00       1109.00
Std Dev        0.00         0.00        0.00         0.00          0.00
95 Prct        0.00      5112.00      108.00       397.00       1109.00
Total             0        10224         216          794          2218
Count             2

%EPC-I-RPQU_BAD_95, 95 Prct for events with counts under 1000 are less precise

%EPC-I-NOEND, 1 Start Event Records had no matching End


30-APR-1990 14:19                 Summary Report                     Page 3
Selection: JUST_ATM                                           DECtrace V1.0-0

Event:  WITHDRAW_EVENT  In Facility:  ATM_SAMPLE      Version:  V1.0

          Elapsed  BUFFERED IO      CPU TIME  CURREN    DIRECT IO   PAGEFAULTS
                                              T PRIO

Minimum      2.34           16             1       9            0            0
Maximum      2.34           16             1       9            0            0
Mean         2.33        16.00          1.00    9.00         0.00         0.00
Std Dev      0.00         0.00          0.00    0.00         0.00         0.00
95 Prct      2.33        16.00          1.00    9.00         0.00         0.00
Total        2.33           16             1       9            0            0
Count           1

          PAGEFAULT      VIRTUAL    GLOBAL WS   PRIVATE WS      WORKING
                IOs         SIZE                                SET SIZ

Minimum           0         5112         108          397          1109
Maximum           0         5112         108          397          1109
Mean           0.00      5112.00      108.00       397.00       1109.00
Std Dev        0.00         0.00        0.00         0.00          0.00
95 Prct        0.00      5112.00      108.00       397.00       1109.00
Total             0         5112         108          397          1109
Count             1

%EPC-I-RPQU_BAD_95, 95 Prct for events with counts under 1000 are less precise


30-APR-1990 14:19                     Index                          Page 4
Selection: JUST_ATM                                           DECtrace V1.0-0

Report Index

Facility Name                 Event Name                      Page

ATM_SAMPLE                    BALANCE_EVENT                   1
ATM_SAMPLE                    DEPOSIT_EVENT                   2
ATM_SAMPLE                    WITHDRAW_EVENT                  3
```

4.5 Text Wrapping in Reports

Wrapping is supported on groupable items, groups, headers, and text report items. This section gives an explanation of the rules for each case.

4.5.1 Groupable Items

Groupable items wrap at the last column of the report. The continuation of the text occurs on the next line without any additional indentation. Example 4–7 shows a portion of a report where the groupable item is a file name.

Example 4–7 Wrapping of a Groupable Item

```
27-MAY-1989 21:45            The Longest Files                       Page 1
Selection: ACC_TRENDS                                         DECtrace V1.0-0

Event TRANSACTION  In Facility RDBVMS                    Version V3.1-0
  For Collections 23MAYAM,24MAYAM
  For Nodes MYVAX1, MYVAX2
  With STREAM_ID = 392
  And  STREAM_ID = 393

DB Name: THIS_IS_A_VERY_LONG_DEVICE_NAME_LOGICAL:[DIRECTORY_LEVEL
         _ONE.DIRECTORY_LEVEL_TWO.DIRECTORY_LEVEL_THREE]THE_DATABASE_FILE
         _NAME.RDB

    .
    .
    .
```

4.5.2 Groups

A Group consists of a set of items. If more report items are provided than can be placed horizontally on the Summary Report, they wrap as a subsequent set of rows. Example 4–8 shows how groups wrap in a report.

Example 4–8 Wrapping of Groups

```
27-MAY-1989 21:45       Monday AM Accounting Transactions           Page 1
Selection: ACC_TRENDS                                          DECtrace V1.0-0

Event TRANSACTION  In Facility RDBVMS                     Version V3.1-0
  For Collections 23MAYAM,24MAYAM
  For Nodes MYVAX1, MYVAX2
  With STREAM_ID = 392
  And  STREAM_ID = 393

Client PC: 3578

           Elapsed   CPU TIME  BUFFERED IO  DIRECT IO  PRIVATE WS   GLOBAL WS

Minimum       0.02          2           25        685         325         287
Maximum      10.65        534          945       1052        2056        4912
Count          346

          RUJ File    AIJ File      Locks
             Reads      Writes  Requested
Minimum         65          12         39
Maximum         75          17       1985
Count          346

Client PC: 3580

           Elapsed   CPU TIME  BUFFERED IO  DIRECT IO  PRIVATE WS   GLOBAL WS

    .
    .
    .
```

4.5.3 Headers

Headers that exceed their report item width parameter will wrap. If an underscore or a space exists in the header, then wrapping occurs at that character. Otherwise, wrapping occurs at the last column. Example 4–9 shows a portion of a database transaction report where the headers wrap.

Example 4–9 Wrapping of Headers

```
27-MAY-1989 21:45       Monday AM Accounting Transactions            Page 1
Selection: ACC_TRENDS                                           DECtrace V1.0-0

Event TRANSACTION  In Facility RDBVMS                      Version V3.1-0
  For Collections 23MAYAM,24MAYAM
  For Nodes MYVAX1, MYVAX2
  With STREAM_ID = 392
  And  STREAM_ID = 393

Client PC: 3578

         Pagefault  Pagefault  Free_VM    Lock Reqs  WS Size    WS Global
                 s        _IO    Bytes
Minimum  . . .
Maximum  . . .
Count    . . .
   .
   .
   .
```

4.5.4 Report Item Text

The text for the Detail Report items are wrapped at the last column of the report item. Example 4–10 shows a portion of a report where item text wraps.

Example 4–10 Wrapping of Report Item Text

```
27-MAY-1989 21:45         Accounting Database Binds                  Page 1
Selection: ACC_TRENDS                                           DECtrace V1.0-0

Event DATABASE  In Facility RDBVMS                         Version V3.1-0
  In Collections 23MAYAM,24MAYAM
  For Nodes MYVAX1, MYVAX2

Point Timestamp           DB Name       Stream Id  Client PC

20-Apr-1989 09:22:45.67  ACC_DISK:[DA        3785     126457
                         TABASES]ACCO
                         UNTING_MAIN.
                         RDB
20-Apr-1989 18:34:23.62  ACC_DISK:[DA        3782     129381
                         TABASES]ACCO
                         UNTING_NEW.R
                         DB
   .
   .
   .
```

4.6 Special Cases for Reporting on Duration Events

When generating a Summary Report for a duration event that contains an item of type level found in both the start and end events, the values associated with the item in the end event are used. If the item is collected only for the start or end event, then the data is picked from the corresponding start or end event.

For Detail and Summary Reports containing duration events, if an item is defined to be of type counter but does not exist in the item list for both the start and end events, then the fields in the report are filled with dashes (-) to indicate an error in the facility definition.

For Summary or Detail Reports, if a starting time-stamp for an event has a value greater than the corresponding end event time-stamp, an error is flagged. For Detail Reports, the elapsed time field is filled with dashes. For Summary Reports, the time-stamps are ignored. The end of the Summary Report for the event contains a message stating the number of occurrences of end time-stamps preceding start time-stamps. Note, if this occurs it might be because the system clock was changed while the collection was running. Another cause may be placing your EPC$START_EVENT service routine call at the end of an event and the EPC$END_EVENT call at the beginning. If the clock is changed, all other items will be correct (for example, if CPU usage was collected, it will be accurate). If the calls are reversed, then the other item values for that event pair will either be zero or negative.

If the report field width of an item is too small to contain a given value, the field is filled with asterisks (*). This condition can be fixed by specifying a wider field using the ITEM report option. For example:

```
$  COLLECT REPORT MY_DATABASE /OPTIONS
Option> EVENT MY_EVENT /FACILITY=MY_FACILITY
Option> ITEM BIG_VALUE /WIDTH=32
Option> CTRL/Z
$
```

If an arithmetic overflow occurs during report generation, the field is filled with percentage signs (%). Data is stored internally using D-floating format. For standard deviation and 95th percentile, the intermediate values may get too large to fit in a D-floating variable (even though the final result would fit).

You can subdivide a report using the /GROUP_BY qualifier to reduce the possibility of an overflow. In this case, the grand total displayed at the end of the report will still contain overflowed values. Another option is to restrict the amount of data reported by using the RESTRICTION option.

If you display a quadword data item as a decimal type, a round-off error may occur. Quadwords displayed as decimal are stored internally as D-floating point. The precision is only 16 decimal-digits with a range of .29*10**-38 to 1.7*10**38. The recommended method is to use the /RADIX=HEXADECIMAL qualifier to the ITEM option on the CREATE DEFINITION command.

4.7 Report Optimization

The following recommendations apply to improving the performance of generating reports from large formatted databases.

- You can put your database journal file on another device by assigning the logical: RDMS$RUJ. For example:

  ```
  $ DEFINE RDMS$RUJ DISK5:[TEMPFILES]
  ```

- You can put your SORT work files on another device by assigning the logicals: SORTWORK0 and SORTWORK1. For example:

  ```
  $ ASSIGN DISK1: SORTWORK0
  $ ASSIGN DISK2: SORTWORK1
  ```

- You may need to increase the enqueue limit (ENQLM) quota for your process. The ENQLM quota allows you to limit the number of locks that a process may take out. Digital recommends a minimum ENQLM value of 1800. If this value is too low, you will get the following error:

  ```
  %EPC-E-Bugcheck, Fatal error
  %NONAME-F-NOMGG,.............
  -RDMS-F-EXQUOTA, exceeded Quota
  -SYSTEM-F-EXENQLM, exceeded Enque Quota
  ```

  To generate a report from a 300k block formatted database, you may need to raise your ENQLM to 10000. Note that increasing the ENQLM has no negative effects on the system in terms of preallocated resources. Also note that you will have to log out and back in again for the change to take affect. See the *VMS Authorize Utility Manual* for information on changing process quotas.

4.8 Report Options

This section describes the report options that you can specify to generate customized reports. Specify options in an options file (or at the DECtrace Option> prompt) when using the /OPTIONS qualifier to the REPORT command. Specify each option on a separate line. You can specify each option more than once. Report options include:

- EVENT
- ITEM
- RESTRICTION

The EVENT option defines characteristics about a specific event. The ITEM and RESTRICTION options define additional characteristics for the EVENT option that immediately precedes them. For example, the following command generates a report from data in the ATM_DATA formatted database. Of all the data in the file, only data collected from processes on node MYVAX1 is reported. Furthermore, only occurrences of the DEPOSIT_EVENT between 1:00 and 3:00, and of the WITHDRAW_EVENT between 2:00 and 4:00 are reported:

```
$  COLLECT REPORT ATM_DATA.RDB /TYPE=SUMMARY -
_$  /STATISTICS=ALL /OPTIONS
Option>  EVENT DEPOSIT_EVENT -
Option>  /SINCE="09-MAY-1990 13:00" -
Option>  /BEFORE="09-MAY-1990 15:00"
Option>  RESTRICTION NODE MYVAX1
Option>  EVENT WITHDRAW_EVENT -
Option>  /SINCE="09-MAY-1990 14:00" -
Option>  /BEFORE="09-MAY-1990 16:00"
Option>  RESTRICTION NODE MYVAX1
Option>  EXIT
$
```

The *Using DECtrace with Digital Products* manual describes how to generate customized reports based on individual events in ALL–IN–1, ACMS, RDB /VMS, and VAX DBMS.

REPORT Options—EVENT

Defines the characteristics of a report on a specific event. You specify this command using the /OPTIONS qualifier to the REPORT command.

Format

EVENT event-name

Command Qualifiers	Defaults
/BEFORE="time"	All data in file
/FACILITY=facility-name	None
/GROUP_BY=(item-name[, . . .])	See text
/INTERVAL=time-unit	See text
/[NO]ITEMS=(item-name[, . . .])	/ITEMS=*
/SINCE="time"	All data in file
/STATISTICS=(stat-type[, . . .])	See text
/SUBTITLE="text-string"	/SUBTITLE=""
/TYPE=report-type	See text

Parameters

event-name
The name of an event to report on.

Qualifiers

/BEFORE="dd-mmm-yyyy hh:mm:ss"
Specifies that DECtrace reports on all data collected at or before the indicated time. By default, DECtrace reports on all data in the formatted database for this event.

If you use the /BEFORE qualifier, you must specify the time as an absolute date and time using a quoted string. You cannot use VMS delta time.

/FACILITY=facility-name
Specifies the name of the facility that contains the event.

/GROUP_BY=(item-name[, . . .])
Specifies the item or items by which to group event occurrences. In SUMMARY and FREQUENCY reports, the report statistics are divided into groups based on equivalent values of the specified items. The final group displays grand total statistics for the entire report.

In DETAIL reports, event occurrences are listed in ascending order based on the item values. If the event is a duration event and the item is of type LEVEL, the end value is used, otherwise, the start value is used.

/INTERVAL=time-unit
Specifies the time interval for FREQUENCY reports. FREQUENCY reports calculate the number of occurrences of an event or events at given time intervals during data collection. The /INTERVAL qualifier specifies the time interval at which event occurrences are calculated. The time interval must be one second, one minute, or one hour specified as one of the following keywords:

SECONDS
MINUTES
HOURS

If an /INTERVAL qualifier is specified on the REPORT command line, the command line qualifier specifies the default interval. The /INTERVAL qualifier on an EVENT option overrides the default interval for this event only.

The default interval is one minute.

The /INTERVAL qualifier is valid for FREQUENCY reports only. The qualifier is ignored for other types of reports.

/ITEMS=(item-name[, . . .])
/NOITEMS
Specifies the item or items on which to report. The /ITEMS qualifier is valid for DETAIL and SUMMARY reports only; it is ignored for FREQUENCY reports.

/SINCE="dd-mmm-yyyy hh:mm:ss"
Specifies that DECtrace reports on all data collected for this event at or after the specified time. By default, DECtrace reports on all the collected data in the formatted database for this event name.

If you use the /SINCE qualifier, you must specify the time as an absolute date and time using a quoted string. You cannot use a VMS delta time.

/STATISTICS=(stat-type[, . . .])
Specifies the summary statistics to include for each item. Valid statistics are:

95_PERCENTILE
ALL
COUNT (default)
MAXIMUM
MEAN
MINIMUM
STANDARD_DEVIATION
TOTAL

If you specify the /STATISTICS qualifier on the REPORT command line, the command line qualifier specifies the default statistics. The /STATISTICS qualifier on an EVENT option overrides the default statistics for this event only.

The /STATISTICS qualifier is valid for SUMMARY reports only. DECtrace ignores the qualifier for DETAIL and FREQUENCY reports.

/SUBTITLE="text-string"
Specifies a subtitle for the report. The subtitle is displayed only at the top of the first page of the report. The subtitle is a text string that can contain any printable characters. The string must be enclosed with quotation marks (" ")

/TYPE=report-type
Specifies the type of report to create. The valid report types are:

DETAIL
FREQUENCY
SUMMARY

If you specify the /TYPE qualifier on the REPORT command line, the command line qualifier specifies the default report type. The /TYPE qualifier on an EVENT option overrides the default report type for this event only.

The default report type is SUMMARY.

Description

Specify report options either in an options file or at the Option> prompt. You must use the /OPTIONS qualifier to the REPORT command to specify options. Specify each option on a separate line. You can specify each option more than once.

REPORT Options—ITEM

Defines the display characteristics of items in the report. You specify this command using the /OPTIONS qualifier to the REPORT command.

Display characteristics include:

- Report heading
- Display width

The default display characteristics of data items are defined in the facility definition. The ITEM report option allows you to customize these characteristics for an item pertaining to the preceding EVENT report option. The characteristics defined in an ITEM option remain in effect only for a single EVENT report option.

Format

ITEM item-name

Command Qualifiers	Defaults
/REPORT_HEADER=text	See text
/WIDTH=number-of-columns	See text

Parameters

item-name
The name of the item for which to define the characteristics.

Qualifiers

/REPORT_HEADER=text
Specifies the text to display as a heading in the report when listing data for the specified item.

The facility definition specifies the default report heading.

/WIDTH=number-of-columns
Specifies the number of columns to reserve for displaying data for the specified item.

The facility definition specifies the default width.

REPORT Options—RESTRICTION

Defines a subset of the data to report on for the preceding EVENT report option. You specify this command using the /OPTIONS qualifier to the REPORT command.

You can selectively report on data based on the following items:

- Collection names
- Image names
- Item values
- Node names
- Process IDs (EPID)

Format

```
               ⎧ COLLECTION collection-name [, . . . ] ⎫
               ⎪ EPID process-ID [, . . . ]            ⎪
RESTRICTION    ⎨ IMAGE image-name [, . . . ]           ⎬
               ⎪ ITEM item-name item-value             ⎪
               ⎩ NODE node-name [, . . . ]             ⎭
```

Parameters

collection-name
The name of one or more collections to use to select data for reporting. Only data collected during the specified collections is reported.

image-name
The name of the executable image which registered with the DECtrace Registrar. Only data collected from processes running the specified images is reported.

The image name that you specify is used as a wildcard in searching through the formatted database for the desired images. The effect is that the restriction is done on **image-name**.

item-name
The name of an item to use to select data for reporting.

item-value
The value of the item to use to select data. Only records that match the item name and value are reported.

node-name
The name of one or more network nodes to use to select data for reporting. Only data collected on the specified nodes is reported.

process-ID
The EPIDs of one or more processes to use to select data for reporting. Only data collected from the specified processes is reported.

5 Instrumenting Applications

DECtrace service routine calls must be **instrumented** in the source program of an application for data collection to take place. The instrumentation consists of adding DECtrace routine calls which identify the program as a DECtrace facility, define the start and end of duration events and the occurrence of point events within the program, and set and delete context for various threads in a multi-threaded environment.

Instrumenting source code is the first step (see Figure 1–1) for application programmers using DECtrace. General users do not need to add anything to their applications to collect data from existing facilities.

Table 5–1 lists all of the DECtrace routine calls used for instrumenting applications. Chapter 8 describes each of the service routines in detail.

Table 5–1 DECtrace Service Routines

Routine Name[1]	Description
EPC$DELETE_CONTEXT	Deletes the context associated with a particular thread (for multi-threaded facilities only).
EPC$END_EVENT	Records the end of a duration event to the data collection file.
EPC$END_EVENTW	Records the end of a duration event to the data collection file and waits for processing to complete before returning.

[1]The service routines are prefaced with "EPC" because that is the registered facility name of DECtrace.

(continued on next page)

Table 5–1 (Cont.)　DECtrace Service Routines

Routine Name[1]	Description
EPC$EVENT	Records the occurrence of a point event to the data collection file.
EPC$EVENTW	Records the occurrence of a point event to the data collection file and waits for processing to complete before returning.
EPC$INIT	Registers a facility with DECtrace to enable data collection on this facility.
EPC$SET_CONTEXT	Sets the context for a new or existing thread so resource utilization items can be collected (for multi-threaded facilities only).
EPC$START_EVENT	Records the start of a duration event to the data collection file.
EPC$START_EVENTW	Records the start of a duration event to the data collection file and waits for processing to complete before returning.

[1]The service routines are prefaced with "EPC" because that is the registered facility name of DECtrace.

5.1 Programming Interface

DECtrace routines must be placed in the source program of a facility in order for data collection to take place for that facility. DECtrace provides routines that allow you to record events which occur during the execution of your application program (executable image.) These routines collect data for each facility in an application program. DECtrace records the data in a data collection file that can be common to one or many different processes that are also collecting data on your system.

Figure 1–1 illustrates that the application program is made up of one or many facilities. In this case the application consists of your own server code (which may or may not be instrumented with DECtrace calls), ACMS, and an application database (probably an Rdb/VMS or a VAX DBMS database). Each of these component facilities is responsible for issuing calls to the DECtrace service routines in order to collect data. The service routines record data in a data collection file that can be shared by other processes that are collecting data on the system or VAXcluster. Note that your server code does not need to have any DECtrace calls in it in order to collect the ACMS event data. A user can schedule data collection from any subset of the facilities defined in the DECtrace administration database, regardless of what application actually called the facility.

Each facility in the application image must include a call to EPC$INIT at the first point of execution of the facility in order to register the facility with DECtrace. Note that each facility needs to call EPC$INIT only once for each image activation of the facility. Then, as the predefined events within the facility execute, the facility places calls to EPC$EVENT(W) and/or to EPC$START_EVENT(W) and EPC$END_EVENT(W) to denote the events from which data is collected.

Events are classified as either **duration events** or **point events**. A duration event has a definite start and end associated with it. A point event simply occurs has no logical start or end. DECtrace provides routines to collect data on either type of event. The EPC$EVENT(W) routine is called for point events. The EPC$START_EVENT(W) routine is called at the start of duration events, and the EPC$END_EVENT(W) routine is called at the end of duration events.

A facility can supply data that contains information specific to the facility (referred to as facility-specific items) on each event call. In addition, DECtrace can collect a set of resource utilization items on behalf of the facility for each event that occurs. Table 5–2 describes the items that comprise this group of items. If a multi-threaded facility (a multi-threaded server, for instance) wants to collect resource utilization items on a per-thread basis, the facility additionally needs to issue calls to EPC$SET_CONTEXT between context switches to ensure that DECtrace accounts the resource utilization items to the correct thread. In order for DECtrace to do this accounting properly, the facility must issue a call to the EPC$SET_CONTEXT service to denote the start of a new thread and also to denote a context change between threads. The EPC$DELETE_CONTEXT service must also be called by the facility to specify that data collection for a thread has ended.

Note *Only multi-threaded facilities need to issue calls to EPC$SET_CONTEXT and EPC$DELETE_CONTEXT.*

Table 5–2 describes the standard resource utilization items collected by DECtrace.

Table 5–2 Resource Utilization Items

Item Name	Item ID Number	Description	Data Type	Usage
BIO	101	Number of buffered I/O operations	Longword	Counter
DIO	102	Number of direct I/O operations	Longword	Counter
PAGEFAULTS	103	Total number of hard and soft page faults	Longword	Counter

(continued on next page)

DECLIT AA VAX PB90A

DECtrace for VMS user's guide

Table 5–2 (Cont.) Resource Utilization Items

Item Name	Item ID Number	Description	Data Type	Usage
PAGEFAULT_IO	104	Number of hard page faults (that is, page faults to or from the disk)	Longword	Counter
CPU	105	Total amount of CPU time in tens of milliseconds.	Longword	Counter
CURRENT_PRIO	106	Current priority of the process	Word	Level
VIRTUAL_SIZE	107	Number of virtual pages that are currently mapped for the process	Longword	Level
WS_SIZE	107	Current working set size of the process	Longword	Level
WS_PRIVATE	109	Number of pages in the working set that are private to the process	Longword	Level
WS_GLOBAL	110	Number of pages in the working set that are globally shared among processes on the system	Longword	Level

Note *DECtrace stamps each data collection record with the current date and time, process identification number (EPID), facility number, and event identifier. Therefore, the elapsed time information for the EPID, the facility that logged the event, and the event identifier are always collected.*

5.2 DECtrace Data Structures

There are two major data structures that the facility can use to detect which events and items to collect: the event flags and the item flags list. The DECtrace event flags (see the EPC$INIT routine in Chapter 8 for a diagram) consist of a list of 128 longword Boolean values. Each element of the list corresponds to a particular event for a facility. If an element is set, it indicates that data should be collected for that event.

The items to be collected for the event are represented in the item flags list data structure. The item flags list is a list of 128-bit elements. Each bit in the list element corresponds to an item identifier (defined in the facility definition). If an item identifier is set, that item is collected for a particular event. The event identifier is used as the offset into this list to obtain the corresponding item flags for an event. In order for the facility to obtain the event flags and item flags list, the facility must allocate virtual memory (either statically or dynamically) and pass the address of these structures in the call to the EPC$INIT service routine.

A buffer, referred to as an **event record**, can be optionally supplied by the client to the EPC$EVENT(W), EPC$START_EVENT(W), and EPC$END_EVENT(W) routines. This buffer contains the facility-specific items for the event. When this buffer is specified, its layout is the same whether the facility collects one item or all items. Any extra items passed by the facility for an event are removed from the data collection files during the merge and format operations, so that only those items that were specified in the facility selection appear in the formatted database.

5.3 Determining Your Events

To effectively use DECtrace with your application, you must correctly identify the events in your program. In addition, you must have a purpose or goal for using DECtrace. Your purpose determines the types of items that you should collect from your events. The ability to select specific groups of events and items at collection time means that you can have multiple goals for your instrumentation.

In the automated teller machine (ATM) sample application, events were chosen based on the usage of the program. A duration event occurs when a customer checks the account balance, or makes a withdrawal or deposit to an account. A different type of event is based on the actual coding of the application. You could create an event for each subroutine or procedure call. This difference in events is based on a difference in the purpose of the instrumentation.

The instrumentation of the ATM application is designed to gather information about the user interface. The goal is to be able to answer questions such as:

- How long does it take to complete a transaction?
- Do customers check their balance before or after every transaction?
- Could overdrafts be reduced or eliminated by displaying the balance on the withdrawal display?
- Are ATM machines used primarily for deposits or withdrawals?

Similar functionality-oriented events which could be developed for the ATM application include:

- Which ATMs are used most often?
- What are the most common withdrawal amounts on each day of the week? $10 on Mondays? $25 on Thursdays? $100 on Fridays?
- Does anyone ever make deposits in the the ATM machine in the shopping mall?

If the goal of instrumenting the ATM application was to improve the performance of the program, then a different set of events could be defined. The events based on functionality could be broken down into more specific events. For example, the withdrawal event consists of three main steps:

1. Get the withdrawal amount.
2. Subtract the amount from the customer's balance.
3. Update the database with the new balance.

You can further break the withdrawal event into steps which detail the actual execution sequence of the program. In fact, the following list of events is similar to the pseudo-code and flowchart used to create the original, non-instrumented, application program.

1. Display the withdrawal screen.
2. Read the withdrawal amount.
3. Find the customer's current balance in the database.
4. Check for possible overdraft.
5. Subtract the withdrawal amount from customer's balance.
6. Lock and then update the customer's account record.

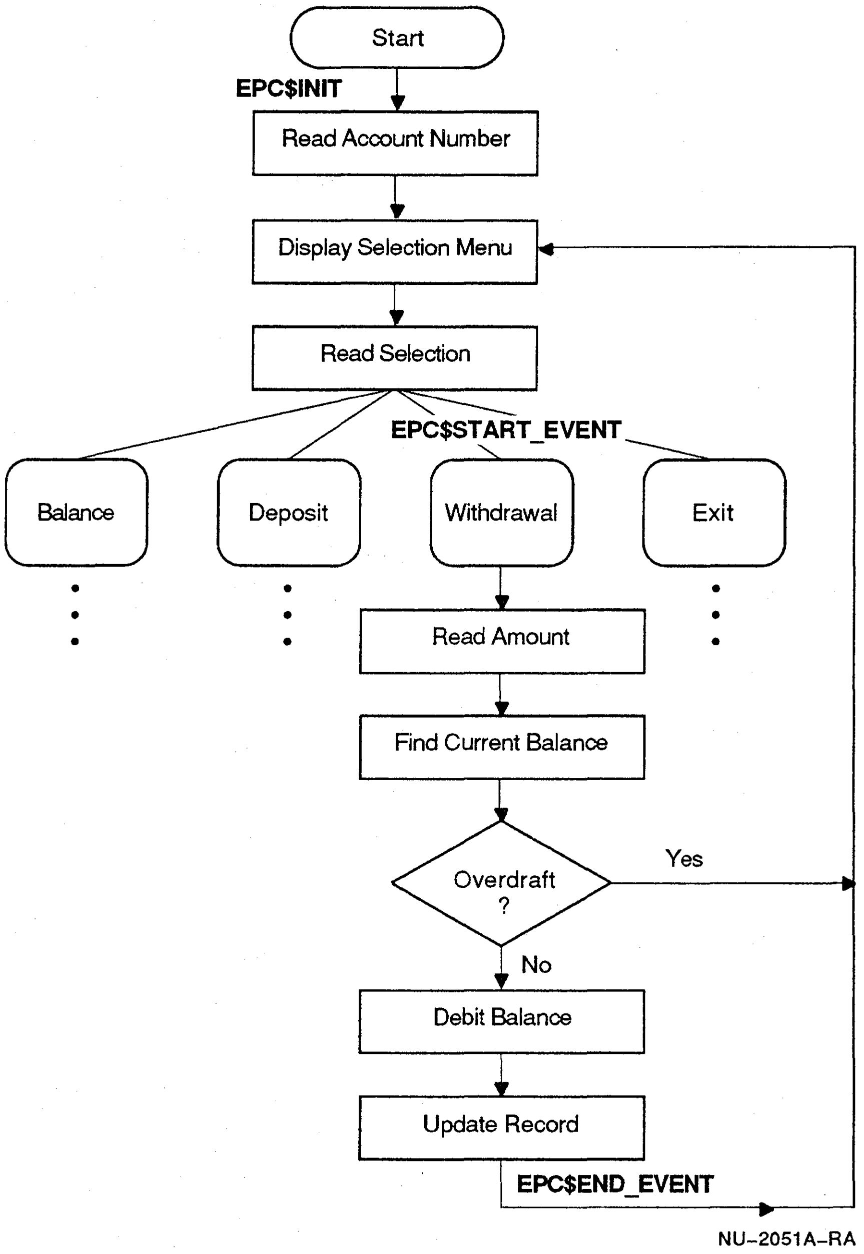
After you have determined what events you want to define for your application, you need to locate the events in a functional diagram of the application. This helps you to locate the actual location of the event in your source code. Figure 5–1 shows a flowchart of the sample ATM application with one event outlined in detail.

5.4 Using EPC$INIT to Register a Facility

The instrumentation of a program begins with an EPC$INIT call. This call should be done early in the execution of a program, because until the DECtrace Registrar process receives this message, no event data can be collected from the application. The EPC$INIT call basically says, "Here I am."

The EPC$INIT call contains information describing the facility. It contains the unique facility ID number, the version of the facility, a list of the events defined for the facility, and the items associated with each event. Optionally, the EPC$INIT call can include a facility-specific **registration ID** which can be used at collection time to distinguish between images using the same facilities. For example, ACMS provides the ACMS application name as the registration ID. See Section 3.1.2 for a description of the uses of the registration ID.

Figure 5–1 Flowchart of the Sample Application Showing One Event



Example 5–1 shows the implementation of the EPC$INIT call in the ATM sample application. The COBOL source file[1] of the ATM application is available in EPC$EXAMPLES:EPC$ATM-SAMPLE.COB.

Example 5–1 Instrumentation of EPC$INIT in the ATM Application

```
IDENTIFICATION DIVISION.
PROGRAM-ID. ATM-SAMPLE.
AUTHOR. Digital Equipment Corporation
*++
    .
    .
    .
*
*The following variables are used for the DECtrace service routines.
*

77 Max_events             PIC 9(9) comp value 128.
77 Facility_number        PIC 9(9) comp value 4094.
77 Errors_event_id        PIC 9(9) comp value 1.
77 Balance_event_id       PIC 9(9) comp value 2.
77 Deposit_event_id       PIC 9(9) comp value 3.
77 Withdrawal_event_id    PIC 9(9) comp value 4.

01 Facility_version       PIC X(4) value "V1.0".
02 Event_flags            PIC 9(9) comp occurs 128 times.
01 Event_handle           PIC 9(9) comp.
01 Cond_status            PIC 9(9) comp.
01 Registration_id        PIC X(15) value "ATM APPLICATION".

    .
    .
    .
PROCEDURE DIVISION.

INITIALIZE-DECTRACE.

*
* Perform facility initialization tasks, including the EPC$INIT call.
*

    CALL "EPC$INIT" USING BY VALUE 0,
                          BY VALUE FACILITY_NUMBER,
                          BY DESCRIPTOR FACILITY_VERSION,
                          BY DESCRIPTOR REGISTRATION_ID,
                          BY REFERENCE EVENT_FLAGS_LIST
                    GIVING COND_STATUS.
```

See Chapter 8 for a detailed description of the format of the EPC$INIT call.

[1] Additional versions of the ATM program, written in FORTRAN and Pascal, are also available in EPC$EXAMPLES.

5.4.1 Waiting for EPC$INIT to Complete

The EPC$INIT call registers a facility with the DECtrace Registrar process. The Registrar records the fact that this process has registered and also compares it to the criteria of any active collections. Although the processing completes quickly, the application program continues to execute, and some events might not get collected.

DECtrace does not cause the application to stop and wait for the Registrar process. However, the application programmer has the ability to make the program wait until data collection is enabled for the process, by passing the event flags and the item flags buffers to the EPC$INIT routine. These buffers are filled asynchronously when the process receives the "Start collecting" message from the Registrar process.

If you do not want to miss any events, poll the event flags and wait until they get written before continuing program execution. Note that a timeout or retry limit should be set because if no active collections are defined to collect data from that image, or if the DECtrace Registrar process is not active, the program will wait indefinitely.

Example 5–2 shows a VAX C code segment from the DECtrace IVP program that waits until the process has registered before continuing program execution.

Example 5–2 Code to Guarantee Collection of All Events

```
/* Wait for the collection to be activated for this     */
/* process. Keep checking the 1st event collection flag */
/* to see if it's set. */
 while((loop_count++ < RETRY_TIMES) && (evntflgs[0] != 1))
  {
    status = lib$wait(&delay_time);
    if (status != SS$_NORMAL)
      {
        printf("Error returned from synchronization after
        call to EPC$INIT\n");
        return (status);
      }
  }
```

5.5 Coding an Event in the ATM Sample Application

Figure 5–1 illustrates the functionality of the ATM WITHDRAWAL event. The instrumentation of the event consists of adding an EPC$START_EVENT call to the beginning of the function, and an EPC$END_EVENT call to the end of the function.

Example 5–3 shows the COBOL code required to implement the WITHDRAWAL event.

Example 5–3 Instrumentation of the WITHDRAWAL Event in the ATM Application

```
* The Withdrawal transaction begins here.
*

START-WITHDRAWAL.

*
* Start of event4: WITHDRAW. For efficiency, test to see if the event is to
* be collected prior to calling EPC$START_EVENTW.
*
    IF EVENT_FLAGS(WITHDRAW_EVENT) = 1
    THEN
      CALL "EPC$START_EVENTW" USING BY VALUE 0,
                                    BY VALUE FACILITY_NUMBER,
                                    BY VALUE WITHDRAW_EVENT,
                                    BY REFERENCE EVENT_HANDLE
                              GIVING COND_STATUS
    END-IF.

*
* Clear the lower portion of the screen and display a withdrawal form.
*
    DISPLAY Withdrawal_heading reversed AT LINE 5 COLUMN 32
                  ERASE TO END OF SCREEN.

    DISPLAY "Amount of withdrawal:" AT LINE 12 COLUMN 15.

    DISPLAY withdrawal-instruction AT LINE 22 COLUMN 1.

PROMPT-FOR-WITHDRAWAL.

*
* Clear the message line and then prompt for the user's input.
*

    DISPLAY SPACE       AT LINE 20 COLUMN 1
                        ERASE TO END OF LINE.

    ACCEPT transaction-amount   BOLD
                                PROTECTED
                                WITH CONVERSION
                                FROM LINE 12 COLUMN 40
                                ERASE TO END OF LINE.
```

(continued on next page)

Example 5–3 (Cont.) Instrumentation of the WITHDRAWAL Event in the ATM Application

```
*
* Find the customer's record and make the withdrawal.
*
    MOVE transaction-failed TO error-message.
    READ DATA-FILE KEY IS Account-ID
        INVALID KEY
                PERFORM Invalid-input THROUGH End-Invalid-Input
                GO TO END-WITHDRAWAL.

    SUBTRACT Transaction-Amount FROM Balance GIVING Balance.

*
* If the withdrawal is larger than the current balance issue a error message.
* Otherwise, update the file.
*
    IF Balance < 0
    THEN MOVE overdraft TO error-message
         PERFORM Invalid-input THROUGH End-Invalid-Input
    ELSE REWRITE Account-record
            INVALID KEY
                PERFORM Invalid-input THROUGH End-Invalid-Input
                GO TO END-WITHDRAWAL
    END-IF.

*
* End of event4: WITHDRAW. For efficiency, test to see if the event is to
* be collected prior to calling EPC$END_EVENTW.
*
    IF EVENT_FLAGS(WITHDRAW_EVENT) = 1
    THEN
      CALL "EPC$END_EVENTW" USING BY VALUE 0,
                                  BY VALUE FACILITY_NUMBER,
                                  BY VALUE WITHDRAW_EVENT,
                                  BY REFERENCE EVENT_HANDLE
                            GIVING COND_STATUS
    END-IF.

END-WITHDRAWAL.
    EXIT.
```

5.6 Instrumenting an Application in VAX C

Instrumenting an application and creating its facility definition (as described in Chapter 6) are similar exercises that can be developed simultaneously. The following sections contain examples of source code instrumented with DECtrace calls and written in VAX C.

5.6.1 Instrumenting Simple Events

The /EVENTS qualifier to the CREATE DEFINITION command specifies a list of events that occur within the facility. For example, the following command defines a facility named DB with three events:

```
!
! DB Facility Definition (Three Events)
!
CREATE DEFINITION DB 2048 /VERSION="V1.0-0"-
    /EVENTS=(DATABASE,TRANSACTION,REQUEST_ACTUAL)
```

These three events would have *only* the standard resource utilization items collected for them by DECtrace. DATABASE would have an event ID of 1, TRANSACTION would have an event ID of 2, and so forth.

For efficiency, the DB facility can use the event IDs to determine whether a facility selection is collecting data for any particular event. This test reduces the overhead of DECtrace on your system when the event does not need to be collected. You eliminate pushing the arguments on the stack and relying on DECtrace to do argument validation. For example, the following source code defines the DB events:

```
#define DATA_EVENT_ID    1          /* DATABASE event ID */
#define TRANS_EVENT_ID   2          /* TRANSACTION event ID */
#define REQ_ACT_EVENT_ID 3          /* REQUEST_ACTUAL event ID */
```

If data is being collected for the TRANSACTION event, DECtrace sets a flag in a data structure called the **event flags list**, the address of which is one of the parameters passed to the EPC$INIT routine.

The event flags list consists of 128 longword Boolean values. For example:

```
static int s_event_flags[128];         /* Event Flags list */
```

The DB facility uses the event ID as an offset to test the appropriate element of the event flags list before calling EPC$EVENT(W) or EPC$START_EVENT(W). For example:

```
if (s_event_flags[TRANS_EVENT_ID-1] != FALSE)
  {
              .
              .
              .
    status = epc$event( . . . )
              .
              .
              .
  }
```

Event ID numbering begins at 1. C array element numbering begins at 0. Thus, an adjustment to the array subscript is necessary.

5.6.2 Instrumenting Events and Items

The /OPTIONS qualifier to the CREATE DEFINITION command provides capabilities beyond those provided by the /EVENTS qualifier, particularly the ability to collect facility-specific items. Table 5–3 summarizes the **facility definition options**, which are described in detail in Section 6.5.

Table 5–3 Summary of Facility Definition Options

Option	Description
ITEM	Binds the name of a facility-specific item to a unique numeric identifier and specifies the characteristics of the data associated with that item.
GROUP	Conveniently allows you to refer to a set of items by a single name. You can use the group name in an EVENT or CLASS option to refer to all of the items within that group.
EVENT	Binds the name of an event to a unique numeric identifier and specifies the items associated with that event.
CLASS	Binds a name to a set of events and a set of items to each event. A facility selection can specify a class name to limit data collection. Digital recommends that application product developers define one or more of the following standard classes: ■ CAPACITY_PLANNING—Includes those events and items that are useful for capacity planning purposes. ■ DEBUGGING—Includes those events and items that are useful for tracing the execution of your application. ■ ERROR_LOGGING—Includes those events and items that are associated with error or exception handling routines in your application. ■ PERFORMANCE—Includes those events and items that are useful for application and/or database tuning. ■ WORKLOAD—Includes those events and items that are useful for gathering information useful for tracing the actual workload of the application and/or database management system.
DEFAULT_CLASS	Indicates which class to collect if none is specified by the facility selection.

5.6.2.1 Collecting Basic Facility-Specific Items The ITEM option defines an item of data that is specific to a facility. It binds the name of an item to a unique numeric identifier and specifies the characteristics of the data associated with that item. The EVENT option specifies the items associated with an event.

For example, the sample database facility definition might contain the following options:

```
!
! DB Facility Definition (Two Items and One Event)
!
CREATE DEFINITION DB 2048 /VERSION="V1.0-0" /OPTIONS
!
ITEM TRANSACTION_ID LONGWORD /ID=1 /REPORT_HEADER="Transaction ID"-
                    /USAGE_TYPE=LEVEL /CHARACTERISTICS=PRINTABLE
!
ITEM PERF_ITEM LONGWORD /ID=2 /REPORT_HEADER="Perf item"-
               /USAGE_TYPE=COUNTER /CHARACTERISTICS=PRINTABLE
!
EVENT TRANSACTION /ID=1 /REPORT_HEADER="Transaction"-
    /ITEMS=(TRANSACTION_ID, PERF_ITEM, RESOURCE_ITEMS)
!
EXIT
```

This facility definition specifies that the event TRANSACTION collects two facility-specific items (TRANSACTION_ID and PERF_ITEM) along with the standard resource utilization items.

The data structure used by the facility to pass facility-specific items to DECtrace is called the **record buffer**. The DB facility uses a record buffer with two fields, one for each facility-specific item:

```
typedef struct record_buffer_s
  {
    int transaction_id; /* Item number 1 */
    int perf_item;      /* Item number 2 */
  } RECORD_BUFFER_T;
          .
          .
          .
    RECORD_BUFFER_T  record_buffer;
```

A record buffer variable must be passed by descriptor. Thus the DB facility contains:

```
#include <descrip.h>            /* VMS Descriptors */
          .
          .
          .
    struct dsc$descriptor_s
                     record_desc;  /* Descriptor for record buffer */
          .
          .
          .
    record_desc.dsc$b_class = DSC$K_CLASS_S;
    record_desc.dsc$w_length = sizeof(RECORD_BUFFER_T);
    record_desc.dsc$a_pointer = &record_buffer;
```

The facility writes the data into the record buffer and calls the appropriate DECtrace service routine to execute the TRANSACTION event, passing the address of the descriptor. For example:

```
record_buffer.transaction_id =  . . .  ;
record_buffer.perf_item =  . . .  ;
status = epc$event( . . .  &record_desc . . .  )
```

A complete sample facility is shown in Example 5–4

Example 5–4   DB Sample Facility

```
/********************************************************************************
*
*  Facility:    DB - Database Facility
*
*  Module:      DB$TRANS.C
*
*  Abstract:    Module handles transactions for use as a code example
*               for issuing DECtrace service routine calls.
*
********************************************************************************/

/********************************************************************************
**      Header Files
********************************************************************************/
#include <stdio.h>              /* C standard I/O                               */
#include <ctype.h>              /* C type library                               */
#include <stdlib.h>             /* C standard library                           */
#include <descrip.h>            /* VMS Descriptors                              */

/********************************************************************************
**      Macro definitions.
********************************************************************************/
#define FAC_NO 2048             /* Facility number */
#define TRANS_EVENT_ID 1        /* Transaction event ID */
```

(continued on next page)

```
/****************************************************************************
**      Type definitions.
****************************************************************************/

typedef struct item_flags_s
  {
    int offset1 : 32;
    int offset2 : 32;
    int offset3 : 32;
    int offset4 : 4;
    int epc_bio_item : 1;              /* DECtrace BIO item number 101 */
    int epc_rest : 27;                 /* DECtrace items 102-128 */
  } ITEM_FLAGS_T;

typedef struct record_buffer_s
  {
    int transaction_id; /* Item number 1 */
    int perf_item;      /* Item number 2 */
  } RECORD_BUFFER_T;

/****************************************************************************
**      Variable declarations.
****************************************************************************/

static int s_event_flags[128];       /* Event Flags list */
static ITEM_FLAGS_T s_event_item_flags[128];
                                     /* Event item flags list */

static $DESCRIPTOR(s_fac_ver,"V1.0-0");

/****************************************************************************
**      External References
****************************************************************************/
globalvalue EPC$_NOTINSTALL;         /* DECtrace not installed message */
globalvalue EPC$_SUCCESS;            /* DECtrace success status */
                                     /* Facility version string */
/****************************************************************************
**      Forward References
****************************************************************************/
int db$$main(void);                  /* Main program entry point         */
int transaction(void);               /* Transaction function       */

/***************************************************************************
*
*  Main Program.
*
***************************************************************************/
int db$main ()
main_program
  {

    int status;           /* Status Code */
```

(continued on next page)

```
    /* Register with DECtrace */
    status = epc$init(1,                        /* EFN */
                      FAC_NO,                   /* Facility number */
                      &s_fac_ver,               /* Facility ver. string */
                      0,                        /* Registration ID string */
                      &s_event_flags,           /* Event flags structure */
                      &s_event_item_flags,      /* Event item flags structure */
                      0,                        /* IOSB */
                      0,                        /* AST address */
                      0);                       /* AST parameter */

    if ((status != EPC$_SUCCESS) && (status != EPC$_NOTINSTALL))
      {
        return (status);
      }

    status = transaction();

    return (TRUE);

  }; /* End db$main */

/******************************************************************************
*
*  Transaction function.
*
******************************************************************************/
int transaction (void)
  {

    int                 event_handle;  /* Event handle */
    RECORD_BUFFER_T     record_buffer;
                                       /* The record buffer containing items for */
                                       /* the transaction event */
    struct dsc$descriptor_s
                        record_desc;   /* Descriptor for record buffer */

    int                 status;        /* Status Code */

    record_desc.dsc$b_class = DSC$K_CLASS_S;
    record_desc.dsc$w_length = sizeof(RECORD_BUFFER_T);
    record_desc.dsc$a_pointer = &record_buffer;
```

(continued on next page)

```
    /* Start of the TRANSACTION event */

    /* Check to see if the TRANSACTION start event should be collected      */
    if (s_event_flags[TRANS_EVENT_ID-1] != FALSE)
      {
        /* Get the items for the transaction event */
        status = get_items(&record_buffer);
        /* Call EPC$START_EVENT to log the start of the transaction event */
        status = epc$start_event(
                                     1,               /* EFN */
                                     FAC_NO,          /* Facility number */
                                     TRANS_EVENT_ID,/* Transaction event ID */
                                     &event_handle, /* Event handle */
                                     0,               /* No thread context var */
                                     &record_desc,  /* Facility items rec. buffer */
                                     0,               /* Completion status */
                                     0,               /* AST address */
                                     0);              /* AST parameter */

        if ((status != EPC$_SUCCESS) && (status != EPC$_NOTINSTALL))
          {
            return (status);
          }

      } /* End Then */

    /* End of the TRANSACTION event */

    /* Check to see if the TRANSACTION end event should be collected */
    if (s_event_flags[TRANS_EVENT_ID-1] != FALSE)
      {
        /* Get the items for the transaction event */
        status = get_items(&record_buffer);
        /* Call EPC$END_EVENT to log the end of the transaction event */
        status = epc$end_event(
                                     1,               /* EFN */
                                     FAC_NO,          /* Facility number */
                                     TRANS_EVENT_ID,/* Transaction event ID */
                                     &event_handle, /* Event handle from start_event */
                                     0,               /* No thread context var */
                                     &record_desc,  /* Facility items rec. buffer */
                                     0,               /* Completion status */
                                     0,               /* AST address */
                                     0);              /* AST parameter */

        if ((status != EPC$_SUCCESS) && (status != EPC$_NOTINSTALL))
          {
            return (status);
          }

      } /* End Then */

    return (TRUE);

  }; /* End transaction function */
```

5.6.2.2 Collecting Facility-Specific Items Efficiently Not all of the items associated with an event are necessarily being collected at any given time. A facility selection can specify a class that collects only a subset of the items in a facility definition. For example, the DB facility definition might contain the following CLASS options where WORKLOAD collects only one facility-specific item:

```
!
! DB Facility Definition (Two Items, One Event, and Two Classes)
!
CREATE DEFINITION DB 2048 /VERSION="V1.0-0" /OPTIONS
!
ITEM TRANSACTION_ID LONGWORD /ID=1 /REPORT_HEADER="Transaction ID"-
                    /USAGE_TYPE=LEVEL /CHARACTERISTICS=PRINTABLE
!
ITEM PERF_ITEM LONGWORD /ID=2 /REPORT_HEADER="Perf item"-
               /USAGE_TYPE=COUNTER /CHARACTERISTICS=PRINTABLE
!
EVENT TRANSACTION /ID=1 /REPORT_HEADER="Transaction"-
    /ITEMS=(TRANSACTION_ID, PERF_ITEM, RESOURCE_ITEMS)-
!
CLASS WORKLOAD TRANSACTION /ITEMS=(TRANSACTION_ID)
!
CLASS PERFORMANCE TRANSACTION /ITEMS=*
!
DEFAULT_CLASS PERFORMANCE
```

Thus, for efficiency, a facility can check each item and pass only those items that are being collected. That information is provided by DECtrace in a data structure called the **item flag list**, which is a two-dimensional (128 by 128 element) bit array. Like the event flags list, the address of the item flags structure is one of the parameters passed to EPC$INIT:

```
typedef struct item_flags_s
  {
    int db_trans_id_item : 1;
    int db_perf_item : 1;
    int offset1 : 30;
    int offset2 : 32;
    int offset3 : 32;
    int offset4 : 32;
  } ITEM_FLAGS_T;

static ITEM_FLAGS_T s_event_item_flags[128];
```

Using the event ID as an offset, the facility examines the appropriate fields in the items flags structure to test the flag for each facility-specific item. If the flag is set, the facility writes the data into the record buffer. For example:

```
if (s_event_flags[TRANS_EVENT_ID-1] != FALSE)
  {
            .
            .
            .
    /* set up descriptor */
           .
           .
           .
  if (s_event_item_flags[TRANS_EVENT_ID-1].db_trans_id_item != FALSE)
     {
     record_buffer.transaction_id =  . . .  ;
     }
  if (s_event_item_flags[TRANS_EVENT_ID-1].db_perf_item != FALSE)
     {
     record_buffer.perf_item =  . . .  ;
     }
           .
           .
           .
  status = epc$event( . . .  &record_desc . . .  )
           .
           .
           .
  }
```

5.6.2.3 Collecting Facility-Specific Items with Maximum Efficiency As stated in Section 5.6.2.2, not all of the items associated with an event are necessarily being collected at any given time. Some facility-specific items require a duration event; others require only a point event. Thus, a facility can minimize overhead by executing an event as either a duration event or a point event, as needed. For example, a facility definition might contain:

```
!
! DB Facility Definition (Two Items, One Elaborated Event, and
! Two Classes)
!
CREATE DEFINITION DB 2048 /VERSION="V1.0-0" /OPTIONS
!
ITEM TRANSACTION_ID LONGWORD /ID=1 /REPORT_HEADER="Transaction ID"-
                    /USAGE_TYPE=LEVEL /CHARACTERISTICS=PRINTABLE
!
ITEM PERF_ITEM LONGWORD /ID=2 /REPORT_HEADER="Perf item"-
               /USAGE_TYPE=COUNTER /CHARACTERISTICS=PRINTABLE
!
EVENT TRANSACTION /ID=1 /REPORT_HEADER="Transaction"-
    /ITEMS=         (TRANSACTION_ID,PERF_ITEM,RESOURCE_ITEMS)-
    /START_EVENT=   (TRANSACTION_ID,PERF_ITEM,RESOURCE_ITEMS)-
    /END_EVENT=     (               PERF_ITEM,RESOURCE_ITEMS)-
    /POINT_EVENT=   (TRANSACTION_ID                         )
!
CLASS WORKLOAD TRANSACTION /ITEMS=(TRANSACTION_ID)
!
CLASS PERFORMANCE TRANSACTION /ITEMS=*
!
DEFAULT_CLASS PERFORMANCE
```

In this facility definition, the TRANSACTION event can be either a duration event or a point event, depending on which items are being collected. The /START_EVENT, /END_EVENT, and /POINT_EVENT qualifiers specify which items are collected in each case.

The PERFORMANCE class requires all items, thus a duration event is needed. However, because the TRANSACTION_ID item does not change during the event, it is only necessary to collect it at the start of the event. The WORKLOAD class requires only the TRANSACTION_ID item; thus a point event is sufficient.

The facility determines whether a duration event or a point event is needed by testing whether an item needed only in one type of event is being collected. In the example, only duration events collect resource utilization items. Thus, the facility can test a resource utilization item and, if that item is being collected, execute a duration event; otherwise a point event. For example, the item flags structure looks something like the following:

```
typedef struct item_flags_s
  {
    int db_trans_id_item : 1;    /* facility specific item 1 */
    int db_perf_item : 1;        /* facility specific item 2 */
    int offset1 : 30;
    int offset2 : 32;
    int offset3 : 32;
    int offset4 : 4;
    int epc_bio_item : 1;        /* DECtrace BIO item number 101 */
    int epc_rest : 27;
  } ITEM_FLAGS_T;
```

In the item flags structure, the DECtrace resource utilization items begin at number 101. Thus, the facility can examine item 101, which happens to be buffered input/output. If that item is being collected, the facility calls EPC$START_EVENT event, otherwise EPC$EVENT. For example:

```
/* Check to see if the TRANSACTION event should be collected */
if (s_event_flags[TRANS_EVENT_ID-1] != FALSE)
  {
    if (s_event_item_flags[TRANS_EVENT_ID-1].epc_bio_item != FALSE)
      {
        /* This transaction is a duration event */
             .
             .
             .
        status = epc$start_event( . . .  )
             .
             .
             .
      }
    else
      {
        /* This transaction is a point event */
             .
             .
             .
       status = epc$event( . . .  )
             .
             .
             .
      }
  }
```

After processing the transaction, the facility can check the item flags structure again and call EPC$END_EVENT if necessary. For example:

```
if ((s_event_flags[TRANS_EVENT_ID-1] != FALSE) &&
    (s_event_item_flags[TRANS_EVENT_ID-1].epc_bio_item != FALSE))
  {
    /* This transaction is a duration event */
              .
              .
              .
    status = epc$end_event( . . .  )
              .
              .
              .
  }
```

For maximum efficiency, the facility can use several different record buffers, each optimized for one type of event. For example:

```
typedef struct s_record_buffer_s /* start event */
  {
    int transaction_id; /* Item number 1 */
    int perf_item;      /* Item number 2 */
  } S_RECORD_BUFFER_T;

typedef struct e_record_buffer_s /* end event */
  {
    int perf_item;      /* Item number 2 */
  } E_RECORD_BUFFER_T;

typedef struct p_record_buffer_s /* point event */
  {
    int transaction_id; /* Item number 1 */
  } P_RECORD_BUFFER_T;
```

When the TRANSACTION event is a point event, the only data that the facility needs to get is the TRANSACTION_ID; thus, the facility sets up and passes a record buffer containing that field only. It does no harm to pass a buffer containing PERF_ITEM as well, but this utilizes more disk space.

When the TRANSACTION event is a duration event, the TRANSACTION_ID item only needs to be collected once (it does not change during the transaction). Thus, the record buffer passed to EPC$END_EVENT contains only one field for PERF_ITEM.

A complete sample facility with maximum performance optimizations is shown in Example 5–5.

Example 5–5 DB Sample Facility—Performance Optimized

```
/******************************************************************************
*
*  Facility:     DB - Database Facility
*
*  Module:       DB$TRANS.C
*
*  Abstract:     Module handles transactions for use as a code example
*                for issuing DECtrace service routine calls.  This code example
*                illustrates the most optimal way of collecting events.
*
******************************************************************************

/******************************************************************************
**      Header Files
*****************************************************************************/
#include <stdio.h>             /* C standard I/O                             */
#include <ctype.h>             /* C type library                             */
#include <stdlib.h>            /* C standard library                         */
#include <descrip.h>           /* VMS Descriptors                            */

/******************************************************************************
**      Macro definitions.
*****************************************************************************/
#define FAC_NO 2048            /* Facility number */
#define TRANS_EVENT_ID 1       /* Transaction event ID */

/******************************************************************************
**      Type definitions.
*****************************************************************************/

typedef struct item_flags_s
  {
    int offset1 : 32;
    int offset2 : 32;
    int offset3 : 32;
    int offset4 : 4;
    int epc_bio_item : 1;              /* DECtrace BIO item number 101 */
    int epc_rest : 27;
  } ITEM_FLAGS_T;

typedef struct e_record_buffer_s
  {
    int perf_item;      /* Item number 2 */
  } E_RECORD_BUFFER_T;

typedef struct p_record_buffer_s
  {
    int transaction_id; /* Item number 1 */
  } P_RECORD_BUFFER_T;

typedef struct s_record_buffer_s
  {
    int transaction_id; /* Item number 1 */
    int perf_item;      /* Item number 2 */
  } S_RECORD_BUFFER_T;
```

(continued on next page)

```
/*****************************************************************************
**      Variable declarations.
*****************************************************************************/

static int s_event_flags[128];        /* Event Flags list */
static ITEM_FLAGS_T s_event_item_flags[128];
                                      /* Event item flags list */

static $DESCRIPTOR(s_fac_ver,"V1.0-0");

/*****************************************************************************
**      External References
*****************************************************************************/
globalvalue EPC$_NOTINSTALL;          /* DECtrace not installed message */
globalvalue EPC$_SUCCESS;             /* DECtrace success status */
                                      /* Facility version string */
/*****************************************************************************
**      Forward References
*****************************************************************************/
int db$$main(void);                   /* Main program entry point         */
int transaction(void);                /* Transaction function       */

/****************************************************************************
*
*  Main program.
*
****************************************************************************/
int db$main ()
main_program
  {

    int status;           /* Status Code */

    /* Register with DECtrace */
    status = epc$init(1,                      /* EFN */
                      FAC_NO,                 /* Facility number */
                      &s_fac_ver,             /* Facility ver. string */
                      0,                      /* Registration ID string */
                      &s_event_flags,         /* Event flags structure */
                      &s_event_item_flags,    /* Event item flags structure */
                      0,                      /* IOSB */
                      0,                      /* AST address */
                      0);                     /* AST parameter */

    if ((status != EPC$_SUCCESS) && (status != EPC$_NOTINSTALL))
      {
        return (status);
      }

    status = transaction();

    return (TRUE);

  }; /* End of db$main */
```

(continued on next page)

```
/*****************************************************************************
*
*  Transaction function.
*
*****************************************************************************/
int transaction (void)
  {

    int                 event_handle; /* Event handle */
    E_RECORD_BUFFER_T e_record_buffer;
                                      /* The record buffer containing items for */
                                      /* the transaction end duration event */
    P_RECORD_BUFFER_T p_record_buffer;
                                      /* The record buffer containing items for */
                                      /* the transaction point event */
    S_RECORD_BUFFER_T s_record_buffer;
                                      /* The record buffer containing items for */
                                      /* the transaction start duration event */
    struct dsc$descriptor_s
                        record_desc;  /* Descriptor for record buffer */

    int                 status;       /* Status Code */

    /* Start of the TRANSACTION event */

    /* Check to see if the TRANSACTION event should be collected */
    if (s_event_flags[TRANS_EVENT_ID-1] != FALSE)
      {
        if (s_event_item_flags[TRANS_EVENT_ID-1].epc_bio_item != FALSE)
          {
            /* This transaction is a duration event */
            record_desc.dsc$b_class = DSC$K_CLASS_S;
            record_desc.dsc$w_length = sizeof(S_RECORD_BUFFER_T);
            record_desc.dsc$a_pointer = &s_record_buffer;
            /* Get the items for the transaction event */
            status = get_items(&s_record_buffer);
            /* Call EPC$START_EVENT for the start of the transaction event */
            status = epc$start_event(
                            1,               /* EFN */
                            FAC_NO,          /* Facility number */
                            TRANS_EVENT_ID,/* Transaction event ID */
                            &event_handle, /* Event handle */
                            0,               /* No thread context var */
                            &record_desc,    /* Facility items rec. buffer */
                            0,               /* Completion status */
                            0,               /* AST address */
                            0);              /* AST parameter */
```

(continued on next page)

```
          if ((status != EPC$_SUCCESS) && (status != EPC$_NOTINSTALL))
            {
              return (status);
            }

        } /* End Then */
      else
        {
          /* This transaction is a point event */
          record_desc.dsc$b_class = DSC$K_CLASS_S;
          record_desc.dsc$w_length = sizeof(P_RECORD_BUFFER_T);
          record_desc.dsc$a_pointer = &p_record_buffer;
          /* Get the transaction ID item for the transaction event */
          status = get_trans_id(&p_record_buffer);
          /* Call EPC$EVENT for the point event */
          status = epc$event(
                           1,              /* EFN */
                           FAC_NO,         /* Facility number */
                           TRANS_EVENT_ID,/* Transaction event ID */
                           0,              /* No thread context var */
                           &record_desc,/* Facility items rec. buffer */
                           0,              /* Completion status */
                           0,              /* AST address */
                           0);             /* AST parameter */

          if ((status != EPC$_SUCCESS) && (status != EPC$_NOTINSTALL))
            {
              return (status);
            }

        } /* End Else */

    } /* End Then */

  /* End of the TRANSACTION event */

  /* Check to see if the end of TRANSACTION event should be collected */
  if ((s_event_flags[TRANS_EVENT_ID-1] != FALSE) &&
      (s_event_item_flags[TRANS_EVENT_ID-1].epc_bio_item != FALSE))
    {
      /* This transaction is a duration event */
      record_desc.dsc$b_class = DSC$K_CLASS_S;
      record_desc.dsc$w_length = sizeof(E_RECORD_BUFFER_T);
      record_desc.dsc$a_pointer = &e_record_buffer;
      /* Get the items for the transaction event */
      status = get_items(&e_record_buffer);
      /* Call EPC$END_EVENT for the end of the transaction event */
      status = epc$end_event(
                           1,              /* EFN */
                           FAC_NO,         /* Facility number */
                           TRANS_EVENT_ID,/* Transaction event ID */
                           &event_handle,  /* Event handle */
                           0,              /* No thread context var */
                           &record_desc,   /* Facility items rec. buffer */
                           0,              /* Completion status */
                           0,              /* AST address */
                           0);             /* AST parameter */
```

(continued on next page)

Example 5–5 (Cont.) DB Sample Facility—Performance Optimized

```
        if ((status != EPC$_SUCCESS) && (status != EPC$_NOTINSTALL))
          {
            return (status);
          }

      } /* End Then */

    return (TRUE);

  }; /* End transaction routine */
```

5.7 Instrumenting a Multi-Threaded Facility Using VAX BLISS-32

This section contains an example of the code modifications required for an event called TRANSACTION for the multi-threaded facility, NEW_TOOL. The example contains code segments written in BLISS-32 and is composed of four routines:

- NEWT$$INITIALIZE
- NEWT$$THREAD_START_RESUME
- NEWT$$THREAD_DELETE
- NEWT$$TRANSACTION

The NEWT$$INITIALIZE routine represents a place in the source code for the facility code where the first code execution takes place. NEWT$$THREAD_START_RESUME represents a place in the code where new threads begin code execution or existing threads resume code execution. The NEWT$$THREAD_DELETE routine represents a place in the code where threads have completed code execution, and NEWT$$TRANSACTION represents a place in the code that comprises a TRANSACTION event.

The following sections describe the steps required to instrument each of the the NEW_TOOL routines with DECtrace service routine calls.

5.7.1 Call EPC$INIT to Register the NEW_TOOL Facility

When the NEW_TOOL facility image activates, this code issues a call to the EPC$INIT service routine. This is the only time that a call to EPC$INIT is issued for the NEW_TOOL facility. The EPC$INIT call registers the facility with DECtrace and allows this facility to collect data, provided that a user schedules a collection to occur while this facility is executing.

```
!-----------------------------------------------------------------------
!
! Global routine NEWT$$INITIALIZE
!
 LITERAL
   max_events = 128,            ! Number of events/facility.
   max_items = 128,             ! Number of items/facility.
   facility_number = 2049,      ! Non-registered facility # for NEW_TOOL
   trans_event_id = 0,          ! Event ID for TRANSACTION event. Note
                                ! that BLISS begins bitvector offsets
                                ! at 0. (Event ID - 1)
   newt$k_dbs_writes_item= 1;   ! Item #2, DBS writes.(Item # - 1 for
                                ! BLISS)  Note that BLISS begins
                                ! bitvector offsets at 0.

 FIELD
   trans_event_rec_fields =     ! TRANSACTION start and end event record
                                ! template.  Note that the record
                                ! contains all possible items for the
                                ! transaction event

   SET
     dbs_reads_item   = [0,0,32,0], ! Total number of database reads.
     dbs_writes_item  = [4,0,32,0], ! Total number of database writes.
   TES;

 OWN
   event_flags : BLOCK [max_events,LONG] VOLATILE,
                                          ! Vector to determine
                                          ! whether to collect data
                                          ! for a given event.
                                          ! Offset is the event ID.
   item_flags : BLOCK [max_events*16,BYTE] VOLATILE,
                                          ! Vector of item_flags.
                                          ! Offset is the event ID.

   trans_event_record_desc : BLOCK[DSC$C_S_BLN,BYTE] PRESET ! Record
                                          ! descriptor
                           ([DSC$W_LENGTH] = 8,! Length of event record
                            [DSC$B_CLASS] = DSC$K_CLASS_S),
                                          ! The pointer will point
                                          ! to the event record in
                                          ! the thread_block.
   cond_status;

BIND
   facility_version = %ASCID 'T1.0-1',    ! The version of the NEW_TOOL
                                          ! facility.

   trans_item_flags =
              item_flags[trans_event_id]-1 : BITVECTOR[max_items];
                                          ! The item flags for the
                                          ! TRANSACTION event.
              .
              .
              .
```

```
!  Perform facility initialization tasks, including a call to the
!  EPC$INIT service.

    cond_status = EPC$INIT (0,facility_number,
                            facility_version,
                            0,event_flags,item_flags);

           .
           .
           .
```

5.7.2 Call EPC$SET_CONTEXT to Set Thread Context

At the place where the thread resumes execution or immediately following the place where the thread block is allocated for a new thread, a call to the EPC$SET_CONTEXT service routine is issued. This call tells DECtrace to keep some new context (resource utilization items) for this new thread and also passes back a new thread identifier to the NEW_TOOL facility. When a thread is resuming execution, this call merely tells DECtrace that the thread context should be updated (that is, resource utilization items should be charged to this thread).

```
!
!----------------------------------------------------------------------
!
!  Global routine called NEWT$$THREAD_START_RESUME
!

!  When creating a new thread, allocate a thread block (called
!  thread_block) for storage local to this thread only.  In addition,
!  initialize the thread_id field to zero, so you pass a zero value
!  thread identifier argument to EPC$SET_CONTEXT.  EPC$SET_CONTEXT
!  then fills in the thread_id field with a unique value associated
!  with this thread.

!  Call EPC$SET_CONTEXT to tell DECtrace that a new thread is
!  executing or that an existing thread is resuming execution.

   cond_status = EPC$SET_CONTEXT (0,thread_block[thread_id]);

           .
           .
           .

!  Call the NEWT$$TRANSACTION routine to process the database
!  transaction.

   cond_status = NEWT$$TRANSACTION (...);

           .
           .
           .
```

5.7.3 Call EPC$DELETE_CONTEXT to Delete the Thread Context

When the thread has completed execution, prior to where the NEW_ TOOL facility deallocates the thread block for this thread, a call to the EPC$DELETE_CONTEXT service routine is issued. The call tells DECtrace no longer keep track of the resource utilization items for this thread.

```
!
!---------------------------------------------------------------------
!
!  Global routine called NEWT$$THREAD_DELETE
!

!  Call EPC$DELETE_CONTEXT to tell DECtrace not to keep resource
!  utilization items for this thread any longer.

   cond_status = EPC$DELETE_CONTEXT (0,thread_block[thread_id]);

!  Deallocate the thread_block.
            .
            .
            .
```

5.7.4 Call EPC$START_EVENT and EPC$END_EVENT to Collect Event Data

At the place in the code where the TRANSACTION event starts, a call to the EPC$START_EVENT service routine is issued. At the place in the code where the TRANSACTION event ends, a call to the EPC$END_EVENT service routine is issued.

```
!
!---------------------------------------------------------------------
!
!  Global routine called NEWT$$TRANSACTION.
!

!  Start of event called TRANSACTION.  For efficiency, test to see
!  if the TRANSACTION event is to be collected prior to calling
!  EPC$START_EVENT.
!
    IF .event_flags[trans_event_id]-1 EQL 1
    THEN

       trans_event_record_desc[DSC$A_POINTER] =
                                thread_block[trans_event_record];
       cond_status = EPC$START_EVENT (0,facility_number,
                                trans_event_id,
                                thread_block[event_handle],
                                thread_block[thread_id],
                                trans_event_record_desc);
            .
            .   } Do all normal processing for the event.
            .
```

```
!
!  Test to see if DBS_WRITES item should be captured for the
!  TRANSACTION event.  If so, when writes occur for the event,
!  update the count in the event record.
!
     IF .trans_item_flags[newt$k_dbs_writes_item]-1 EQL 1
     THEN thread_block[trans_event_record][dbs_writes_item] =
               .thread_block[trans_event_record][dbs_writes_item] + 1;
             .
             .
             .
!
!  End of event called TRANSACTION.  For efficiency, test to see if
!  the TRANSACTION event is to be captured prior to calling
!  EPC$END_EVENT.
!
    IF .event_flags[trans_event_id]-1 EQL 1
    THEN

       trans_event_record_desc[DSC$A_POINTER] =
                                 thread_block[trans_event_record];
       cond_status = EPC$END_EVENT (0,facility_number,
                                 trans_event_id,
                                 thread_block[event_handle],
                                 thread_block[thread_id],
                                 trans_event_record_desc);
```

5.8 Linking an Instrumented Program

There are two methods for linking an instrumented program. Which one you use depends on whether or not DECtrace is installed on the target system (the system where you will be running the application), and the version of VMS on that system.

The DECtrace shareable image (SYS$SHARE:EPC$SHR.EXE) is included with VMS Version 5.2 and higher, and is present on all systems where the DECtrace software is installed. To link an instrumented program on these systems, simply use the VMS LINK command. The resulting executable image will run on any system with VMS Version 5.2 or higher, or any system where DECtrace is installed. The following example shows the command to link the ATM sample application:

```
$  LINK EPC$ATM-SAMPLE.OBJ
```

If you intend to run your application on a VMS Version 5.0 or Version 5.1 system where DECtrace is not installed, you must use LIB$FIND_IMAGE_SYMBOL in your instrumentation to resolve symbolic references to the DECtrace service routines. Use the LIB$FIND_IMAGE_SYMBOL call to conditionalize each of the DECtrace service routine calls. If DECtrace is not installed on the target system, the program should not attempt to make any of the DECtrace calls. Make the calls using the symbols returned from LIB$FIND_IMAGE_SYMBOL instead of making calls directly to the DECtrace service routines. Note that this method also works for VMS Version 5.2 and higher target systems, although it is not required.

Example 5–6 shows a code segment of an instrumented program which uses LIB$FIND_IMAGE_SYMBOL calls. The example is written in VAX C.

Example 5–6 Instrumentation Using LIB$FIND_IMAGE_SYMBOL

```
extern unsigned long int (*MYFAC$EPC$DELETE_CONTEXT)();
extern unsigned long int (*MYFAC$EPC$END_EVENT)();
extern unsigned long int (*MYFAC$EPC$END_EVENTW)();
extern unsigned long int (*MYFAC$EPC$EVENT)();
extern unsigned long int (*MYFAC$EPC$EVENTW)();
extern unsigned long int (*MYFAC$EPC$INIT)();
extern unsigned long int (*MYFAC$EPC$SET_CONTEXT)();
extern unsigned long int (*MYFAC$EPC$START_EVENT)();
extern unsigned long int (*MYFAC$EPC$START_EVENTW)();
    .
    .
    .

$DESCRIPTOR(facility_ver, "V1.0-0"); /* Facility Version */
$DESCRIPTOR (epc_lib_desc , "EPC$SHR");

$DESCRIPTOR (delete_context_desc , "EPC$DELETE_CONTEXT");
$DESCRIPTOR (end_event_desc , "EPC$END_EVENT");
$DESCRIPTOR (end_eventw_desc , "EPC$END_EVENTW");
$DESCRIPTOR (event_desc , "EPC$EVENT");
$DESCRIPTOR (eventw_desc , "EPC$EVENTW");
$DESCRIPTOR (init_desc , "EPC$INIT");
$DESCRIPTOR (set_context_desc , "EPC$SET_CONTEXT");
$DESCRIPTOR (start_event_desc , "EPC$START_EVENT");
$DESCRIPTOR (start_eventw_desc , "EPC$START_EVENTW");


/* Get entry points into DECtrace */
status = LIB$FIND_IMAGE_SYMBOL (&epc_lib_desc ,
                   &delete_context_desc , &MYFAC$EPC$DELETE_CONTEXT);
/* Error check */
status = LIB$FIND_IMAGE_SYMBOL (&epc_lib_desc ,
                   &end_event_desc , &MYFAC$EPC$END_EVENT);
/* Error check */
status = LIB$FIND_IMAGE_SYMBOL (&epc_lib_desc ,
                   &end_eventw_desc , &MYFAC$EPC$END_EVENTW);
/* Error check */
status = LIB$FIND_IMAGE_SYMBOL (&epc_lib_desc ,
                   &event_desc , &MYFAC$EPC$EVENT);
/* Error check */
status = LIB$FIND_IMAGE_SYMBOL (&epc_lib_desc ,
                   &eventw_desc , &MYFAC$EPC$EVENTW);
/* Error check */
status = LIB$FIND_IMAGE_SYMBOL (&epc_lib_desc ,
                   &init_desc , &MYFAC$EPC$INIT);
/* Error check */
status = LIB$FIND_IMAGE_SYMBOL (&epc_lib_desc ,
                   &set_context_desc , &MYFAC$EPC$SET_CONTEXT);
```

(continued on next page)

```
/* Error check */
status = LIB$FIND_IMAGE_SYMBOL (&epc_lib_desc ,
                   &start_event_desc , &MYFAC$EPC$START_EVENT);
/* Error check */
status = LIB$FIND_IMAGE_SYMBOL (&epc_lib_desc ,
                   &start_eventw_desc , &MYFAC$EPC$START_EVENTW);
/* Error check */
/* If all calls to LIB$FIND_IMAGE_SYMBOL returned success */
if ((status & 1) == 1)
  {
     /* Issue initialize call to DECtrace Registrar */

        status = MYFAC$EPC$INIT(event_flag,  /* Event Flag Number      */
                    MYFAC$_FACILITY,         /* Facility number        */
                    &facility_ver,           /* Facility version string*/
                    &registration_id,        /* Registration ID string */
                    &MYFAC$_EPC_EVENT_FLAGS, /* Event flags structure  */
                    &MYFAC$_EPC_ITEM_FLAGS,  /* Event item flags struc */
                    &init_iosb,              /* IOSB                   */
                    0,                       /* AST address            */
                    0);                      /* AST parameter          */
  }
```

6 Creating Facility Definitions

Each application program that is instrumented with DECtrace service routine calls must provide a **facility definition**, which describes the *events* in the application and the *items* to collect for each event. A facility definition also provides useful information for reporting purposes, such as header information for the events and items.

This chapter describes how to work with DECtrace facility definitions including:

- Creating facility definitions
- Deleting facility definitions
- Transporting facility definitions between systems
- Displaying information about facility definitions stored in the DECtrace administration database

Table 6–1 summarizes the DECtrace commands available to manipulate facility definitions. See Chapter 7 for a description of the commands and their syntax.

Table 6–1 Facility Definition Commands

Command	Description
CREATE DEFINITION	Creates a facility definition and stores it in the DECtrace administration database.
DELETE DEFINITION	Deletes a facility definition from the DECtrace administration database.

(continued on next page)

Table 6–1 (Cont.) Facility Definition Commands

Command	Description
EXTRACT DEFINITION	Extracts a facility definition from the DECtrace administration database and stores it in a binary file.
INSERT DEFINITION	Inserts a facility definition into the DECtrace administration database from a binary DECtrace facility definition file.
SHOW DEFINITION	Displays information about facility definitions in the DECtrace administration database.

6.1 Creating a Facility Definition

DECtrace facility definitions consist of:

- Name of the facility.
- ID number of the facility. Numbers in the range 1 to 2047 are reserved to Digital and are referred to as **registered facilities**. Numbers in the range 2048 to 4095 specify **non-registered facilities**.
- Version of the facility (up to 10 characters). Some typical formats of version strings are:
  - Vnn.nn (versions of software)
  - Vnn.nn-nn (versions and baselevels or revision levels of software)
  - Tnn.nn-nn (field test versions of software and baselevel)
- List of the events that occur within the facility.
- List of facility-specific items associated with the events.

The format of the CREATE DEFINITION command is:

```
                                              [ /EVENTS=(event_name[, . . . ])  ]
CREATE DEFINITION facility_name facility_id   [ /OPTIONS[=file_spec]            ]
                                              [ /REPLACE                        ]
                                              [ /VERSION="version_code"         ]
```

There are two methods for creating a facility definition. You can create a simple definition that associates only the default resource utilization items (shown in Table 6–2) with your events, or you can create an advanced definition with items that are specific to your application.

To collect the default data, you can create the definition interactively. For example, the following command creates the facility definition for the DB facility which has two events:

```
$  COLLECT CREATE DEFINITION DB 2048 /VERSION="V1.0-0" -
_$  /EVENTS=(EVENT_1, EVENT_2)
```

To create a more detailed definition including facility-specific items, you should use the /OPTIONS qualifier to the CREATE DEFINITION command and specify an options file. There are several different ways to use the /OPTIONS qualifier:

- Enter the /OPTIONS qualifier interactively without supplying a file specification; DECtrace prompts you for the options. For example:

  ```
  $  COLLECT
  DECtrace>  CREATE DEFINITION DB 2048 /VERSION="V1.0-0" /OPTIONS
  Option>  ITEM TRANSACTION_ID LONGWORD /ID=1  . . .
      .
      .
      .
  ```

- Enter the /OPTIONS qualifier interactively and supply a file specification that DECtrace uses as a source for facility definition options. For example:

  ```
  $  COLLECT CREATE DEFINITION DB 2048 /VERSION="V1.0-0" -
  _$  /OPTIONS=DB_OPTIONS.OPT
  ```

- Invoke DECtrace to execute a file that contains the CREATE DEFINITION command as well as the options:

  ```
  $  COLLECT @DEFINE_DB.COM
  ```

- Execute a VMS command procedure that invokes DECtrace, enters the CREATE DEFINITION command, and so forth. For example:

  ```
  $  @DB.COM
  ```

6.1.1 Creating and Defining Events

DECtrace collects data for **events** which occur during run time. In the automated teller machine (ATM) sample application, the defined events are:

- Customer checks account balance.
- Customer deposits funds into account.
- Customer withdraws funds from account.
- Customer makes an error and the application displays an error message.

The balance, deposit, and withdrawal events are **duration events**, which begin when the customer selects the appropriate menu option and end when the transaction is complete. The error message event is a **point event**, which has no logical start or end and simply occurs.

Once you have determined the events that your facility contains, you must instrument your source code with DECtrace service routine calls. Chapter 5 describes how to instrument your code, and Chapter 8 describes the formats of the service routines.

After instrumenting your code, you must create a facility definition that includes the events you have implemented. The /EVENTS qualifier to the CREATE DEFINITION command specifies a list of events that occur within the facility. Note that in a simple facility definition, no distinction is made between point and duration events. For example, the following command creates a facility definition for the ATM_SAMPLE application having four events:

```
$ COLLECT CREATE DEFINITION ATM_SAMPLE 4094 /VERSION="V1.0" -
_$ /EVENTS=(ERROR_DISPLAY, BALANCE_EVENT, DEPOSIT_EVENT, WITHDRAW_EVENT)
```

These events would have the resource utilization items shown in Table 6–2 collected for them. To collect facility-specific items or to collect a subset of the resource items, you must use the /OPTIONS qualifier instead of the /EVENTS qualifier. The /OPTIONS qualifier allows you to specify the following EVENT characteristics:

- Name of the event.
- ID number of the event. The default is the next unused event ID between 1 and 128.
- Report header to use for the event.
- List of all items associated with the event.
- Items to collect at the start of duration events.
- Items to collect at the end of duration events.
- Items to collect for point events[1].

The format of the EVENT option is:

```
                      ┌ /IDENTIFIER=event_id                   ┐
                      │ /ITEMS=(item_name[, . . . ])           │
EVENT event_name      │ /START_EVENT=(item_name[, . . . ])     │
                      │ /END_EVENT=(item_name[, . . . ])       │
                      │ /POINT_EVENT=(item_name[, . . . ])     │
                      └ /REPORT_HEADER="text"                  ┘
```

[1] See Section 5.6.2.3 for a description of how some events can be handled as either a point or duration.

The following example creates the ATM_SAMPLE facility definition using the /OPTIONS qualifier:

```
$  COLLECT CREATE DEFINITION ATM_SAMPLE 4094 /VERSION="V1.0" /OPTIONS
Option>  EVENT ERROR_DISPLAY /ID=1 /REPORT_HEADER="Errors" -
_Option>  ITEMS=(RESOURCE_ITEMS) /POINT_EVENT=(RESOURCE_ITEMS)
Option>  EVENT BALANCE_EVENT /ID=2 /REPORT_HEADER="Balance" -
_Option>  /ITEMS=(RESOURCE_ITEMS) /START_EVENT=(RESOURCE_ITEMS) -
_Option>  /END_EVENT=(RESOURCE_ITEMS)
Option>  EVENT DEPOSIT_EVENT /ID=3 /REPORT_HEADER="Deposits" -
_Option>  /ITEMS=(RESOURCE_ITEMS) /START_EVENT=(RESOURCE_ITEMS) -
_Option>  /END_EVENT=(RESOURCE_ITEMS)
Option>  EVENT WITHDRAW_EVENT /ID=4 /REPORT_HEADER="Withdrawals" -
_Option>  /ITEMS=(RESOURCE_ITEMS) /START_EVENT=(RESOURCE_ITEMS) -
_Option>  /END_EVENT=(RESOURCE_ITEMS)
Option>  EXIT
```

Alternately, you could use a command file to create the ATM_SAMPLE facility definition[2]. For example:

```
$  COLLECT @EPC$ATM-FAC-DEF.COM
```

EPC$ATM-FAC-DEF.COM contains the following lines:

```
CREATE DEFINITION ATM_SAMPLE 4094 /VERSION="V1.0" /OPTIONS
!
EVENT ERROR_DISPLAY /ID=1 /REPORT_HEADER="Errors" /ITEMS=(RESOURCE_ITEMS) -
      /POINT_EVENT=(RESOURCE_ITEMS)
EVENT BALANCE_EVENT /ID=2 /REPORT_HEADER="Balance" /ITEMS=(RESOURCE_ITEMS) -
      /START_EVENT=(RESOURCE_ITEMS) /END_EVENT=(RESOURCE_ITEMS)
EVENT DEPOSIT_EVENT /ID=3 /REPORT_HEADER="Deposits" /ITEMS=(RESOURCE_ITEMS)-
      /START_EVENT=(RESOURCE_ITEMS) /END_EVENT=(RESOURCE_ITEMS)
EVENT WITHDRAW_EVENT /ID=4 /REPORT_HEADER="Withdrawals" -
      /ITEMS=(RESOURCE_ITEMS) -
      /START_EVENT=(RESOURCE_ITEMS) /END_EVENT=(RESOURCE_ITEMS)
!
EXIT
```

See Section 6.5 for a full description of the facility definition options.

6.1.2 Creating and Defining Items

Items are elements of data associated with each event. They describe *what* is collected for the event. Table 6–2 describes the set of standard DECtrace resource utilization items. Note that these items are often taken as a whole, and to facilitate this, they can be referred to by the group name: RESOURCE_ITEMS. If disk space is not a concern, collecting all of the resource items requires less CPU overhead than collecting a subset of them.

[2] You can find EPC$ATM-FAC-DEF.COM in EPC$EXAMPLES.

Table 6–2 Standard Resource Utilization Items

Item Name	Item ID Number	Description	Data Type	Usage
BIO	101	Number of buffered I/O operations	Longword	Counter
DIO	102	Number of direct I/O operations	Longword	Counter
PAGEFAULTS	103	Total number of hard and soft page faults	Longword	Counter
PAGEFAULT_IO	104	Number of hard page faults (that is, page faults to or from the disk)	Longword	Counter
CPU	105	Total amount of CPU time in tens of milliseconds	Longword	Counter
CURRENT_PRIO	106	Current priority of the process	Word	Level
VIRTUAL_SIZE	107	Number of virtual pages that are currently mapped for the process	Longword	Level
WS_SIZE	107	Current working set size of the process	Longword	Level
WS_PRIVATE	109	Number of pages in the working set that are private to the process	Longword	Level
WS_GLOBAL	110	Number of pages in the working set that are globally shared among processes on the system	Longword	Level

Note *DECtrace stamps each data collection record with the current date and time, extended process identification number (EPID), facility number, and event identifier. Therefore, the elapsed time information for the EPID, the facility that logged the event, and the event identifier are always collected.*

You can collect information in addition to (or instead of) that provided by the resource utilization items by creating **facility-specific items** and associating them with your events. You use the /OPTIONS qualifier to the CREATE DEFINITION command to create facility-specific items. The /OPTIONS qualifier allows you to specify to following ITEM characteristics:

- Name of the item.
- Data type of the item.
- ID number of the item. The default is the next unused item ID between 1 and 100.
- Maximum size (in bytes) of the item.
- Report header for the item.
- Width of the column when displaying the item in a DECtrace report.

- Usage type of the item. Valid types are: LEVEL, COUNTER, PERCENT, TEXT, and PRIVATE.
- Characteristics of the item. Items can be either printable or nonprintable.

The format of the ITEM option is:

ITEM item_name datatype

/IDENTIFIER=item_id
/SIZE=n-bytes
/REPORT_HEADER="text"
/REPORT_WIDTH=number-of-spaces
/USAGE_TYPE=usage-type
/CHARACTERISTICS=[non]printable

The following example creates a facility definition for the NEW_TOOL facility with five items and two events:

```
! NEW_TOOL facility definition
CREATE DEFINITION NEW_TOOL 2049 /VERSION="T1.0-1" /OPTIONS
!
ITEM STREAM_ID LONGWORD /ID=1 /REPORT_HEADER="Stream Id" /REPORT_WIDTH=11 -
  /USAGE_TYPE=LEVEL /CHARACTERISTICS=PRINTABLE /RADIX=HEXADECIMAL
ITEM DBS_READS LONGWORD /ID=2 /REPORT_HEADER="Data File Reads" -
  /REPORT_WIDTH=10 /USAGE_TYPE=COUNTER /CHARACTERISTICS=PRINTABLE
ITEM DBS_WRITES LONGWORD /ID=3 /REPORT_HEADER="Data File Writes" -
  /REPORT_WIDTH=10 /USAGE_TYPE=COUNTER /CHARACTERISTICS=PRINTABLE
ITEM DB_NAME ASCIC /ID=4 /REPORT_HEADER="DB Name" /REPORT_WIDTH=25 -
  /USAGE_TYPE=TEXT /CHARACTERISTICS=PRINTABLE /SIZE=255
ITEM LOCK_MODE BYTE /ID=5 /REPORT_HEADER="Lock Mode" /REPORT_WIDTH=4 -
  /USAGE_TYPE=LEVEL /CHARACTERISTICS=PRINTABLE
!
EVENT TRANSACTION /ID=1 /REPORT_HEADER="Transaction" -
  /ITEMS=(RESOURCE_ITEMS, STREAM_ID, DB_NAME, LOCK_MODE, DBS_READS, DBS_WRITES) -
  /START_EVENT=(RESOURCE_ITEMS, STREAM_ID, DB_NAME, DBS_READS, DBS_WRITES) -
  /POINT_EVENT=(STREAM_ID, DB_NAME, LOCK_MODE) -
  /END_EVENT=(RESOURCE_ITEMS, STREAM_ID, DB_NAME, DBS_READS, DBS_WRITES)
EVENT DATABASE /ID=2 /REPORT_HEADER="Database" -
  /ITEMS=(RESOURCE_ITEMS, STREAM_ID, DB_NAME, DBS_READS, DBS_WRITES) -
  /START_EVENT=(RESOURCE_ITEMS, STREAM_ID, DB_NAME, DBS_READS, DBS_WRITES) -
  /POINT_EVENT=(STREAM_ID, DB_NAME) -
  /END_EVENT=(RESOURCE_ITEMS, STREAM_ID, DB_NAME, DBS_READS, DBS_WRITES)
!
EXIT
```

Note that the /OPTIONS and /EVENTS (as described in Section 6.1.1) qualifiers are mutually exclusive. See Section 6.5 for a full description of the facility definition options. Section 5.6.2.1 describes how to instrument items into your application source code.

6.1.3 Creating and Defining Collection Classes

DECtrace can collect data from all of the events and items defined for your facility. However, you can reduce overhead in terms of both CPU and disk space utilization by collecting only data that pertains to your current needs.

You can create subsets of your events and items that are relative to specific collection purposes. A facility selection (as described in Chapter 2) can specify a **collection class** to limit data collection to only that data that is important to the user. You can create a class for any usage. However, Digital recommends that application product developers define one or more of the following standard classes:

- CAPACITY_PLANNING—Includes those events and items that are useful for capacity planning purposes.
- DEBUGGING—Includes those events and items that are useful for tracing the execution of your application.
- ERROR_LOGGING—Includes those events and items that are associated with error handling routines in your application.
- PERFORMANCE—Includes those events and items that are useful for application and/or database tuning.
- WORKLOAD—Includes those events and items that are useful for gathering information for tracing the actual workload of the application or the database management system.

Use the /OPTIONS qualifier to the CREATE DEFINITION command to create collection classes. The format of the CLASS option is:

CLASS class_name event_name *ITEMS=(item_name[, . . .])*

To define a collection class, list each event you want to include in the class and specify the items that you want to collect for that event. Later, you can specify that class on the CREATE SELECTION command (see Section 2.2). If you do not include an event in the class definition, no data is collected for that event when the application executes. The following example defines WORKLOAD and PERFORMANCE classes for the NEW_TOOL application:

```
! NEW_TOOL facility definition
CREATE DEFINITION NEW_TOOL 2049 /VERSION="T1.0-1" /OPTIONS
!
ITEM STREAM_ID LONGWORD /ID=1 /REPORT_HEADER="Stream Id" /REPORT_WIDTH=11 -
  /USAGE_TYPE=LEVEL /CHARACTERISTICS=PRINTABLE /RADIX=HEXADECIMAL
ITEM DBS_READS LONGWORD /ID=2 /REPORT_HEADER="Data File Reads" -
  /REPORT_WIDTH=10 /USAGE_TYPE=COUNTER /CHARACTERISTICS=PRINTABLE
ITEM DBS_WRITES LONGWORD /ID=3 /REPORT_HEADER="Data File Writes" -
  /REPORT_WIDTH=10 /USAGE_TYPE=COUNTER /CHARACTERISTICS=PRINTABLE
ITEM DB_NAME ASCIC /ID=4 /REPORT_HEADER="DB Name" /REPORT_WIDTH=25 -
  /USAGE_TYPE=TEXT /CHARACTERISTICS=PRINTABLE /SIZE=255
ITEM LOCK_MODE BYTE /ID=5 /REPORT_HEADER="Lock Mode" /REPORT_WIDTH=4 -
  /USAGE_TYPE=LEVEL /CHARACTERISTICS=PRINTABLE
```

```
!
EVENT TRANSACTION /ID=1 /REPORT_HEADER="Transaction" -
  /ITEMS=(RESOURCE_ITEMS, STREAM_ID, DB_NAME, LOCK_MODE, DBS_READS, DBS_WRITES) -
  /START_EVENT=(RESOURCE_ITEMS, STREAM_ID, DB_NAME, DBS_READS, DBS_WRITES) -
  /POINT_EVENT=(STREAM_ID, DB_NAME, LOCK_MODE) -
  /END_EVENT=(RESOURCE_ITEMS, STREAM_ID, DB_NAME, DBS_READS, DBS_WRITES)
EVENT DATABASE /ID=2 /REPORT_HEADER="Database" -
  /ITEMS=(RESOURCE_ITEMS, STREAM_ID, DB_NAME, DBS_READS, DBS_WRITES) -
  /START_EVENT=(RESOURCE_ITEMS, STREAM_ID, DB_NAME, DBS_READS, DBS_WRITES) -
  /POINT_EVENT=(STREAM_ID, DB_NAME) -
  /END_EVENT=(RESOURCE_ITEMS, STREAM_ID, DB_NAME, DBS_READS, DBS_WRITES)
!
CLASS WORKLOAD DATABASE /ITEMS=(STREAM_ID, DBS_READS, DBS_WRITES)
CLASS WORKLOAD TRANSACTION /ITEMS=(STREAM_ID, LOCK_MODE, DBS_READS, DBS_WRITES)
!
CLASS PERFORMANCE TRANSACTION /ITEMS=*
!
DEFAULT_CLASS WORKLOAD
!
EXIT
```

Note that the ALL class is created by default when you create a facility definition and is composed of all events and associated items in your facility. The ALL class is also the default class unless you specify another with the DEFAULT_CLASS option.

See Section 6.5 for a full description of the facility definition options. Section 5.6.2.2 describes how to use classes efficiently and how to instrument the relevant events and items into your application source code.

6.2 Deleting Facility Definitions

You can delete facility definitions from the DECtrace administration database with the DELETE DEFINITION command. Note that you do not delete the actual facility images from your system, but merely the DECtrace facility definitions. To delete definitions of registered facilities (numbered 1 to 2047), you must have VMS BYPASS or SYSPRV privilege. To delete definitions of non-registered facilities (numbered 2048 to 4095), you must either be the creator of the definition or have VMS BYPASS or SYSPRV privilege.

The following example deletes the facility definition for version T1.0 of the TEMP_PROG facility:

```
$  COLLECT DELETE DEFINITION TEMP_PROG /VERSION="T1.0"
Delete TEMP_PROG T1.0 [N]:  YES
%EPC-S-FACDEL_DELETED, Facility definition TEMP_PROG T1.0 was deleted
```

You cannot delete a facility definition if any active or pending collections are collecting data from that facility. Use the SHOW SELECTION /FORMAT=BRIEF command to determine if a facility has active or pending data collection associated with it. If it does, you must use the CANCEL COLLECTION command to stop data collection before you can delete the facility definition. In addition, you have to delete any facility selections that explicitly specify the version of the facility whose definition you want to delete.

If a facility selection does not specify the version of the facility and another version exists on the system, you can delete the facility definition without canceling scheduled data collection or deleting the facility selection.

The following example shows how to delete a facility definition that has active data collection associated with it:

```
$ COLLECT DELETE DEFINITION TEMP_PROG /VERSION="T1.0" /NOCONFIRM
%EPC-F-FACDEL_NOTDELETED, Facility is referenced by active data collection
$ COLLECT
DECtrace> SHOW SELECTION/FORMAT=BRIEF

22-AUG-1989 12:11       Facility Selection Information              Page 1
                                                            DECtrace V1.0-0

   Selection Name         Facility          Version     Class
   --------------------   ---------------   ----------  --------------------
   A1_DATA_ENTRY          OA                (latest)    ALL
                          TESTER            T4.1        ALL
                          RDBVMS            V3.1        ALL

   TEST_SELECT            TEMP_PROG         T1.0        PERFORMANCE

DECtrace> CANCEL COLLECTION /SELECTION=TEST_SELECT /NOCONFIRM
%EPC-S-SCHED_CANCEL, Collection QUICK_TEST is cancelled

DECtrace> DELETE SELECTION TEST_SELECT /NOCONFIRM
%EPC-S-SELDEL_DELETED, Facility selection TEST_SELECT was deleted
DECtrace> DELETE DEFINITION TEMP_PROG /VERSION="T1.0"
Delete TEMP_PROG T1.0 [N]: YES
%EPC-S-FACDEL_DELETED, Facility definition TEMP_PROG T1.0 was deleted
DECtrace> EXIT
$
```

You can use the SHOW DEFINITION /FORMAT=NAMES_ONLY command to confirm the spelling of the names and the version codes for the facility definitions that you want to delete.

6.3 Transporting Facility Definitions

You can transport a facility definition to another system or VAXcluster with the EXTRACT DEFINITION and INSERT DEFINITION commands. This feature is useful in creating installation kits for your applications. Your installation procedure can automatically add a facility definition to the DECtrace administration database.

6.3.1 Extracting Definitions

You can extract a facility definition from the DECtrace administration database with the EXTRACT DEFINITION command. The command creates a binary file containing the facility definition information for the specified facility. You must have VMS BYPASS, SYSPRV, or READALL privilege to extract registered facility definitions or non-registered facility definitions that were created by another user. The files are stored in binary format to prevent tampering with the facility definitions.

The following command extracts the facility definition for version V1.0 of the ATM_SAMPLE facility. The facility definition is stored in the file ATM_FAC_DEF.EPC$DEF:

```
$  COLLECT EXTRACT DEFINITION ATM_SAMPLE ATM_FAC_DEF /VERSION="V1.0"
%EPC-S-FACEXT, Facility definition(s) was successfully extracted
```

6.3.2 Inserting Definitions Using a KITINSTAL.COM Procedure

DECtrace provides a callback procedure compliant with the VMSINSTAL procedure that any product can use to automatically insert a binary facility definition into the DECtrace administration database. The procedure is called INSERT_DEF, and is located in SYS$UPDATE:EPC$VMSINSTAL_CALLBACK.COM. If DECtrace is not installed on the system, the procedure inserts the facility definition into the DECtrace facility library: SYS$COMMON:[SYSLIB]EPC$FACILITY.TLB, which is shipped with Version 5.2 or higher of the VMS operating system.

During installations of DECtrace, the system manager is asked to run EPC$INSERT.COM (in EPC$EXAMPLES:), which checks the facility library and moves any facility definitions found there into the DECtrace administration database.

Example 6–1 shows a segment of a KITINSTAL.COM procedure which includes the callback. The facility definition for MY_APPLICATION V1.0 is inserted into the DECtrace administration database.

Example 6–1 Sample Procedure to Insert Binary Facility Definitions

```
$ INSERT_DEF:
$ !+
$ !  Attempt to insert the facility definition.  On error (DECtrace not
$ !  installed) insert it into sys$share:epc$facility.tlb
$ !
$ !  Parameters:
$ !
$ !  P2 : Return value (see below)
$ !  P3 : File specification for the binary facility definition file.
$ !  P4 : The facility name, up to 27 characters in length
$ !  P5 : The facility version, string up to 10 characters in length.
$ !
$ !  Return values in P2:
$ !
$ !   I : Successfully inserted into DECtrace admin db.
$ !   L : Successfully inserted into SYS$SHARE:EPC$FACILITY.TLB.
$ !   N : No operations were performed.
```

(continued on next page)

Example 6–1 (Cont.) Sample Procedure to Insert Binary Facility Definitions

```
$ !
$ !-
$ !+
$ !     Example of how to call INSERT_DEF callback.
$ !-
$ !+
$ ! Set up debug mode (VMSINSTAL OPTION K)
$ !-
$     _o = "!"                  ! default is nodebug
$     _x = "!"                  ! default is nodebug
$     IF .NOT. P2 THEN GOTO START
$     _o = "SET VERIFY"         ! ON around OUR code
$     _x = "SET NOVERIFY"       ! OFF around Callbacks
$_o
$
$ START:
$ !+
$ ! Copy the callback procedures to sys$update.
$ !-
$    COPY/NOLOG VMI$KWD:EPC$VMSINSTAL_CALLBACK.COM VMI$ROOT:[SYSUPD]*.*;
$ !
$ INSERT_FACDEFS:
$_x
$    VMI$CALLBACK PRODUCT EPC$VMSINSTAL_CALLBACK:INSERT_DEF -
          RETURN_SYMBOL VMI$KWD:MY_FAC_DEF.EPC$DEF -
          MY_APPLICATION V1.0-0
$_o
$    IF RETURN_SYMBOL .EQS. "N" -
     THEN WRITE SYS$OUTPUT "DECtrace facility definition could not be installed"
$
$ !+
$ ! Delete the callback procedures from sys$update.
$ !-
$    DELETE/NOLOG VMI$ROOT:[SYSUPD]EPC$VMSINSTAL_CALLBACK.COM;*
$ !
```

6.3.3 Inserting Definitions Without KITINSTAL.COM

If your installation procedure is not VMSINSTAL compliant, you can use the INSERT DEFINITION command to insert a facility definition (previously created with the EXTRACT DEFINITION command) into the DECtrace administration database on a target system or VAXcluster. You must have VMS BYPASS or SYSPRV privilege to insert a facility definition for a facility that already exists in the target DECtrace administration database and that was created by another user.

The following command inserts the facility definition for the ATM_SAMPLE facility into the DECtrace administration database on the target system:

```
$ COLLECT INSERT DEFINITION ATM_FAC_DEF.EPC$DEF
%EPC-S-FACCRE, Facility definition ATM_SAMPLE was created
```

Registered facilities insert their facility definitions into the DECtrace administration database as part of their normal installation procedure. However, if DECtrace does not exist on the target system (the INSERT DEFINITION command fails), the facility definitions can be stored in the **DECtrace facility library**. The installation procedure must perform the following steps:

1. Create the facility library SYS$COMMON:[SYSLIB]EPC$FACILITY.TLB if it does not already exist[3]:

   ```
   $ LIBRARY/TEXT/CREATE=(BLOCK=3,HISTORY=32767,KEYSIZE=39) -
   _$  SYS$COMMON:[SYSLIB]EPC$FACILITY.TLB
   ```

2. Use the VMS librarian to insert the binary facility definition file into the DECtrace facility library (note that the module name is a concatenation of the facility name and the version string):

   ```
   $ LIBRARY/TEXT/INSERT/REPLACE/MODULE=ATM_SAMPLEV1.0 -
   _$ SYS$COMMON:[SYSLIB]EPC$FACILITY.TLB ATM_FAC_DEF.EPC$DEF
   ```

During the DECtrace installation, the system manager can run EPC$INSERT.COM (in EPC$EXAMPLES:) to automatically insert all of the facility definitions that exist in SYS$SHARE:EPC$FACILITY.TLB into the administration database.

If you plan to remove DECtrace from your system but want to retain the facility definitions stored in your DECtrace administration database (in case you later decide to reinstall DECtrace), you can extract the facility definitions from your DECtrace administration database and store them in the DECtrace facility library.

Use the /LIBRARY qualifier to the INSERT DEFINITION command to insert a binary facility definition file into the DECtrace facility library. The following example extracts the facility definition for the ATM_SAMPLE facility from the DECtrace administration database and stores it in the facility library:

```
$ COLLECT EXTRACT DEFINITION ATM_SAMPLE ATM_FAC.DEF /VERSION="V1.0"
$ COLLECT INSERT DEFINITION ATM_FAC_DEF.EPC$DEF /LIBRARY
```

6.4 Displaying Facility Definitions

You can display information about the facility definitions stored in the DECtrace administration database using the SHOW DEFINITION command. The command takes one argument: the name of a facility. If you do not specify a facility name, DECtrace displays information on all of the facilities defined on the system.

[3] EPC$FACILITY.TLB is shipped with Version 5.2 or higher of the VMS operating system.

You can specify the amount of information to display about a facility definition by using the FORMAT qualifier to the SHOW DEFINITION command. There are two valid format types:

- FULL
- NAMES_ONLY (default)

The following example writes information in the NAMES_ONLY format about all of the facility definitions on the system to the file FAC_NAMES.DAT:

```
$ COLLECT SHOW SELECTION /FORMAT=NAMES_ONLY /OUTPUT=FAC_NAMES.DAT
```

6.4.1 FULL Format

If you specify /FORMAT=FULL with the SHOW DEFINITION command, DECtrace displays a full description of facility definitions stored in the DECtrace administration database. You can display the description of a single definition if you include its name and version code on the command line. For example, the following command displays the complete description of the ATM_SAMPLE definition:

```
$ COLLECT SHOW DEFINITION ATM_SAMPLE /VERSION="V1.0" /FORMAT=FULL
```

The FULL format display includes the following information:

- Name and version of the facility
- Facility ID number
- Creation date of the facility definition
- User name of the facility definition's creator
- Description of events, items, groups, and classes defined for the facility

Example 6–2 shows a sample of the display produced with the SHOW DEFINITION /FORMAT=FULL command. The "x" in the Position field indicates that the items are default item collected by DECtrace and that the image does not need to allocate space for the item in its event record buffer. See Section 5.2 for a description of DECtrace data structures.

Example 6–2 Display for SHOW DEFINITION Using the Full Format

```
28-AUG-1989 17:02       Facility Definition Information                Page 1
Full Report                                                    DECtrace V1.0-0

Facility:      ATM_SAMPLE
Number:        4094
Version:       V1.0
Creation Date: 25-AUG-89 14:17
Created By:    SYSTEM

Events:

  Event Name       Event ID  Report Header
  ---------------  --------  ----------------
  ERROR_DISPLAY    1         ERROR_DISPLAY
  BALANCE_EVENT    2         BALANCE_EVENT
  DEPOSIT_EVENT    3         DEPOSIT_EVENT
  WITHDRAW_EVENT   4         WITHDRAW_EVENT

Items:

  Item Name        Item  Datatype  Max.  Usage    Item Report   Report  Char Rad
                    ID             Size  Type     Header        Width
  ---------------  ----  --------  ----  -------  ------------  ------  ---- ---
  BIO              101   LONGWORD     4  COUNTER  BUFFERED IO     11    PRT  DEC
  DIO              102   LONGWORD     4  COUNTER  DIRECT IO       11    PRT  DEC
  PAGEFAULTS       103   LONGWORD     4  COUNTER  PAGEFAULTS      11    PRT  DEC
  PAGEFAULT_IO     104   LONGWORD     4  COUNTER  PAGEFAULT       11    PRT  DEC
                                                  IOs
  CPU              105   LONGWORD     4  COUNTER  CPU TIME        11    PRT  DEC
  CURRENT_PRIO     106   WORD         2  LEVEL    CURRENT PRIO    6     PRT  DEC
  VIRTUAL_SIZE     107   LONGWORD     4  LEVEL    VIRTUAL SIZE    11    PRT  DEC
  WS_SIZE          108   LONGWORD     4  LEVEL    WORKING SET     11    PRT  DEC
                                                  SIZ
  WS_PRIVATE       109   LONGWORD     4  LEVEL    PRIVATE WS      11    PRT  DEC
  WS_GLOBAL        110   LONGWORD     4  LEVEL    GLOBAL WS       11    PRT  DEC

Item Groups:

  Item Group Name:  RESOURCE_ITEMS

  Item Name
  -----------------
  BIO
  DIO
  PAGEFAULTS
  PAGEFAULT_IO
  CPU
  CURRENT_PRIO
  VIRTUAL_SIZE
  WS_SIZE
  WS_PRIVATE
  WS_GLOBAL

  Class:  ALL
```

(continued on next page)

Example 6–2 (Cont.) Display for SHOW DEFINITION Using the Full Format

```
Event Name:    BALANCE_EVENT

 Record Type       Item Name        Position
 ---------------   ---------------  --------
 START EVENT       BIO                 x


28-AUG-1989 17:02      Facility Definition Information              Page 2
Full Report                                                  DECtrace V1.0-0

 Record Type       Item Name        Position
 ---------------   ---------------  --------
                   DIO                 x
                   PAGEFAULTS          x
                   PAGEFAULT_IO        x
                   CPU                 x
                   CURRENT_PRIO        x
                   VIRTUAL_SIZE        x
                   WS_SIZE             x
                   WS_PRIVATE          x
                   WS_GLOBAL           x

 END EVENT         BIO                 x
                   DIO                 x
                   PAGEFAULTS          x
                   PAGEFAULT_IO        x
                   CPU                 x
                   CURRENT_PRIO        x
                   VIRTUAL_SIZE        x
                   WS_SIZE             x
                   WS_PRIVATE          x
                   WS_GLOBAL           x

 POINT EVENT       BIO                 x
                   DIO                 x
                   PAGEFAULTS          x
                   PAGEFAULT_IO        x
                   CPU                 x
                   CURRENT_PRIO        x
                   VIRTUAL_SIZE        x
                   WS_SIZE             x
                   WS_PRIVATE          x
                   WS_GLOBAL           x
```

(continued on next page)

Example 6–2 (Cont.) Display for SHOW DEFINITION Using the Full Format

```
Event Name:     DEPOSIT_EVENT

 Record Type        Item Name        Position
 ----------------   ---------------  --------
 START EVENT        BIO                 x
                    DIO                 x
                    PAGEFAULTS          x
                    PAGEFAULT_IO        x
                    CPU                 x
                    CURRENT_PRIO        x
                    VIRTUAL_SIZE        x
                    WS_SIZE             x
                    WS_PRIVATE          x
                    WS_GLOBAL           x

 END EVENT          BIO                 x
                    DIO                 x
                    PAGEFAULTS          x
                    PAGEFAULT_IO        x
                    CPU                 x
                    CURRENT_PRIO        x
                    VIRTUAL_SIZE        x
                    WS_SIZE             x
                    WS_PRIVATE          x


28-AUG-1989 17:02       Facility Definition Information              Page 3
Full Report                                                 DECtrace V1.0-0

 Record Type        Item Name        Position
 ----------------   ---------------  --------
                    WS_GLOBAL           x

 POINT EVENT        BIO                 x
                    DIO                 x
                    PAGEFAULTS          x
                    PAGEFAULT_IO        x
                    CPU                 x
                    CURRENT_PRIO        x
                    VIRTUAL_SIZE        x
                    WS_SIZE             x
                    WS_PRIVATE          x
                    WS_GLOBAL           x
```

(continued on next page)

Example 6–2 (Cont.) Display for SHOW DEFINITION Using the Full Format

```
Event Name:    ERROR_DISPLAY

 Record Type        Item Name         Position
 ----------------   ---------------   --------
 START EVENT        BIO                  x
                    DIO                  x
                    PAGEFAULTS           x
                    PAGEFAULT_IO         x
                    CPU                  x
                    CURRENT_PRIO         x
                    VIRTUAL_SIZE         x
                    WS_SIZE              x
                    WS_PRIVATE           x
                    WS_GLOBAL            x

 END EVENT          BIO                  x
                    DIO                  x
                    PAGEFAULTS           x
                    PAGEFAULT_IO         x
                    CPU                  x
                    CURRENT_PRIO         x
                    VIRTUAL_SIZE         x
                    WS_SIZE              x
                    WS_PRIVATE           x
                    WS_GLOBAL            x

 POINT EVENT        BIO                  x
                    DIO                  x
                    PAGEFAULTS           x
                    PAGEFAULT_IO         x
                    CPU                  x
                    CURRENT_PRIO         x
                    VIRTUAL_SIZE         x
                    WS_SIZE              x
                    WS_PRIVATE           x
                    WS_GLOBAL            x

Event Name:    WITHDRAW_EVENT

 Record Type        Item Name         Position
 ----------------   ---------------   --------
 START EVENT        BIO                  x
                    DIO                  x
```

(continued on next page)

Example 6–2 (Cont.) Display for SHOW DEFINITION Using the Full Format

```
28-AUG-1989 17:02       Facility Definition Information             Page 4
Full Report                                                   DECtrace V1.0-0

 Record Type          Item Name          Position
 ----------------     ---------------    --------
                      PAGEFAULTS            x
                      PAGEFAULT_IO          x
                      CPU                   x
                      CURRENT_PRIO          x
                      VIRTUAL_SIZE          x
                      WS_SIZE               x
                      WS_PRIVATE            x
                      WS_GLOBAL             x

 END EVENT            BIO                   x
                      DIO                   x
                      PAGEFAULTS            x
                      PAGEFAULT_IO          x
                      CPU                   x
                      CURRENT_PRIO          x
                      VIRTUAL_SIZE          x
                      WS_SIZE               x
                      WS_PRIVATE            x
                      WS_GLOBAL             x

 POINT EVENT          BIO                   x
                      DIO                   x
                      PAGEFAULTS            x
                      PAGEFAULT_IO          x
                      CPU                   x
                      CURRENT_PRIO          x
                      VIRTUAL_SIZE          x
                      WS_SIZE               x
                      WS_PRIVATE            x
                      WS_GLOBAL             x
```

6.4.2 NAMES_ONLY Format

If you specify /FORMAT=NAMES_ONLY with the SHOW DEFINITION command, DECtrace lists the names of the facilities defined on your system. This format is useful to determine the latest version of a facility installed on your system. The report also shows the collection classes that are available for each facility. This is useful to check before creating a facility selection (see Section 2.2).

Example 6–3 shows a sample of the display produced with the SHOW DEFINITION /FORMAT=NAMES_ONLY command.

Example 6-3 Display for SHOW DEFINITION Using the Names Only Format

```
25-AUG-1989 14:28       Facility Definition Information              Page 1
Names Only Report                                              DECtrace V1.0-0

 Facility:              Version:     Creation Date:       Class:
 ---------------------  ----------   ------------------   ------------------
 ATM_SAMPLE             V1.0         25-AUG-89 14:17      ALL               (D)
 RDBVMS                 V3.1         20-AUG-89 15:21      ALL
                                                          PERFORMANCE       (D)
                                                          WORKLOAD
```

6.5 Facility Definition Options

The /OPTIONS qualifier to the CREATE DEFINITION command provides capabilities beyond those provided by the /EVENTS qualifier, particularly the ability to collect facility-specific items. Table 6-3 summarizes the **facility definition options**.

Table 6-3 Summary of Facility Definition Options

Option	Description
ITEM	Binds the name of a facility-specific item to a unique numeric identifier and specifies the characteristics of the data associated with that item.
GROUP	Conveniently allows you to refer to a set of items by a single name. You can use the group name in an EVENT or CLASS option to refer to all of the items within that group.
EVENT	Binds the name of an event to a unique numeric identifier and specifies the items associated with that event.

(continued on next page)

Table 6–3 (Cont.) Summary of Facility Definition Options

Option	Description
CLASS	Binds a name to a set of events and a set of items to each event. A facility selection can specify a class name to limit data collection. Digital recommends that application product developers define one or more of the following standard classes: ■ CAPACITY_PLANNING—Includes those events and items that are useful for capacity planning purposes. ■ DEBUGGING—Includes those events and items that are useful for tracing the execution of your application. ■ ERROR_LOGGING—Includes those events and items that are associated with error or exception handling routines in your application. ■ PERFORMANCE—Includes those events and items that are useful for application and/or database tuning. ■ WORKLOAD—Includes those events and items that are useful for gathering information useful for tracing the actual workload of the application and/or database management system.
DEFAULT_CLASS	Indicates which class to collect from if none is specified by the facility selection.

Note *Facility options are order-dependent. Each option must be defined before it is referenced. The order dependencies are: ITEM, GROUP, EVENT, CLASS, and DEFAULT_CLASS.*

ITEM

The ITEM option identifies and describes the characteristics of a data item that the facility can collect.

Format

ITEM item-name datatype

Command Qualifiers	Defaults
/CHARACTERISTICS=(characteristic[, . . .])	/CHARACTERISTICS=PRINTABLE
/IDENTIFIER=item-id	Next unused item-id
/RADIX=base-system	/RADIX=DECIMAL
/REPORT_HEADER=text	/REPORT_HEADER=item-name
/REPORT_WIDTH=number-of-spaces	See text
/SIZE=n-bytes	See text
/USAGE_TYPE=usage-type	None

Parameters

item-name

Specifies a text string that names the item. The string must be a valid VMS name, cannot end with a dollar sign ($) or underscore (_), and cannot be an Rdb/VMS reserved word.

datatype

Specifies the data type of the item. Table 6–4 lists the valid data types.

Table 6–4 ITEM Data Types

Data Type	Description
ASCIC	Varying-length counted string of up to 255 bytes
ASCIW	Varying-length counted string of up to 16,383 bytes
BYTE	Signed byte
FIXED_ASCIC	Fixed-length counted string of up to 255 bytes

(continued on next page)

Table 6–4 (Cont.) ITEM Data Types

Data Type	Description
LONGWORD	Signed longword (4 bytes)
QUADWORD	Signed quadword (8 bytes)
WORD	Signed word (2 bytes)

Qualifiers

/CHARACTERISTICS=(characteristic[, . . .])
Either PRINTABLE or NONPRINTABLE that specifies whether the contents of an item can be displayed. The default is PRINTABLE.

/IDENTIFIER=item-id
An integer between 1 and 100 that specifies a unique identifier for the item. (Identifiers 101 to 128 are reserved for DECtrace-defined items.) The default is the next unused ID between 1 and 100.

/RADIX=base-system
Either HEXADECIMAL or DECIMAL that specifies the base of the number system used for numerical items. The default radix is DECIMAL.

/REPORT_HEADER="text"
A text string that specifies a heading to display when creating reports using this item. The default is the name of the item.

/REPORT_WIDTH=number-of-spaces
An integer that specifies the width of the column to use when displaying item values in a report. The default width depends on the item data type, as listed in Table 6–5.

Table 6–5 ITEM Default Report Widths

Data Type	Width
ASCIC	80 columns
ASCIW	80
BYTE	4
FIXED_ASCIC	80
LONGWORD	11
QUADWORD	32
WORD	6

/SIZE=n-bytes
An integer that specifies the maximum size in bytes of the ASCIC, FIXED_ASCIC, and ASCIW data types. The /SIZE qualifier is required for these data types and ignored for all others.

/USAGE_TYPE=usage-type
One of the keywords in Table 6–6 that specifies how the data is to be used.

Table 6–6 ITEM Usage Types

Usage Type	Description
COUNTER	Typically a running count or total; its value either always increases or always decreases for the duration of the event. In a DECtrace Summary report, for items with usage type COUNTER, DECtrace displays the difference between the start and end values.
LEVEL	A meter or gauge that indicates the current value of some metric; its value may increase or decrease over the duration of the event.
PERCENT	A percentage; similar to LEVEL, except that it has upper and lower limits of 100 and 0, respectively.
PRIVATE	Facility-defined data that does not fall into one of the other usages. DECtrace does not attempt to display this item on DECtrace reports.
TEXT	Text characters.

Valid usage types depend on the data type of the item, as shown in Table 6–7.

Table 6–7 ITEM Usage Types by Data Type

Data Type	Valid Usage Types	Default
ASCIC	TEXT, PRIVATE	TEXT
ASCIW	TEXT, PRIVATE	TEXT
BYTE	COUNTER, LEVEL, PERCENT, PRIVATE	COUNTER
FIXED_ASCIC	TEXT, PRIVATE	TEXT
LONGWORD	COUNTER, LEVEL, PERCENT, PRIVATE	COUNTER
QUADWORD	COUNTER, LEVEL, PERCENT, PRIVATE	COUNTER
WORD	COUNTER, LEVEL, PERCENT, PRIVATE	COUNTER

GROUP

The GROUP option allows you to refer to a set of items by a single name. You can use a group name in the /ITEMS qualifier of an EVENT or CLASS option to refer to all of the items within that group.

Format

GROUP group-name

Command Qualifier	**Default**
/ITEMS=(item-name[, . . .])	None

Parameter

group-name
Specifies the name of the group.

Qualifier

/ITEMS=(item-name[, . . .])
Specifies the items contained in the group. This is a required qualifier.

EVENT

The EVENT option identifies and describes the characteristics of an event including the set of items to collect and (optionally) the items to collect when the event is a point event, start event, or end event. By default, DECtrace places each event in the facility class named ALL.

Format

EVENT event-name

Command Qualifiers	**Defaults**
/END_EVENT=(item-name[, . . .])	See text
/FIRST_SEGMENT_SIZE=n-bytes	None
/IDENTIFIER=event-id	Next unused event ID
/[NO]ITEMS=(item-name[, . . .])	/NOITEMS
/POINT_EVENT=(item-name[, . . .])	See text
/REPORT_HEADER=text	/REPORT_HEADER=event-name
/SEGMENT_SIZE=n-bytes	None
/START_EVENT=(item-name[, . . .])	See text

Parameter

event-name
Specifies the name of the event. The event name must be a valid VMS name and cannot end with a dollar sign ($) or underscore (_).

Qualifiers

/END_EVENT=(item-name[, . . .])
Specifies the items that DECtrace collects at the end of a duration event. If you omit the /END_EVENT qualifier, DECtrace uses the /ITEMS qualifier to define the items to collect at the end of a duration event.

/FIRST_SEGMENT_SIZE=n-bytes
Specifies the size of the first string segment for the event relation in the formatted database. If you do not specify a size for the first segment, the DECtrace formatting component uses 64 bytes.

See the OPTIMIZATION PARAMETERS under the FORMAT command in Chapter 7 and Section A.1.2 for more information about string segmentation.

/IDENTIFIER=item-id
Specifies an integer between 1 and 128 that uniquely identifies the event. The default is the next unused event identifier between 1 and 128.

/ITEMS=(item-name[, . . .])
/NOITEMS (default)
Specifies the set of all items to collect for an event. The default is /NOITEMS. This is a required qualifier.

/POINT_EVENT=(item-name[, . . .])
Specifies the items that DECtrace collects for a point event. If you omit the /POINT_EVENT qualifier, DECtrace uses the /ITEMS qualifier to define the items to collect for a point event.

/REPORT_HEADER=text
Specifies the text to display as a heading when creating reports using this event. The default report heading is the name of the event.

/SEGMENT_SIZE=n-bytes
Specifies the size of the remaining string segments (the first having been defined with the /FIRST_SEGMENT_SIZE qualifier) for the event relation in the formatted database. If you do not specify a size for the segments, the DECtrace formatting component uses 128 bytes.

See the OPTIMIZATION PARAMETERS under the FORMAT command in Chapter 7 and Section A.1.2 for more information about string segmentation.

/START_EVENT=(item-name[, . . .])
Specifies the items that DECtrace collects at the beginning of a duration event. If you omit the /START_EVENT qualifier, DECtrace uses the /ITEMS qualifier to define the items to collect at the beginning of a duration event.

Description

In the /ITEMS, /START_EVENT, /END_EVENT, and /POINT_EVENT qualifiers, items are collected in the order specified. You can specify:

- Any item or item group that has already been defined.
- A predefined resource utilization item.
- The predefined item group RESOURCE_ITEMS, which collects all resource utilization items.
- An asterisk (*) as a wildcard symbol, meaning all of the ITEM options in the facility definition as well as resource utilization items. (This is for the /ITEMS qualifier only.)

CLASS

The CLASS option defines a new class if the specified class does not already exist and binds an event (with a subset of its associated items) to a class.

Format

CLASS class-name event-name

Command Qualifier	**Default**
/[NO]ITEMS=(item-name[, . . .])	/NOITEMS

Parameters

class-name
Specifies the name of the class. A class name must be a unique 1 to 32 character string consisting of alphanumeric characters, dollar signs, and underscores. The string must be unique within the context of the facility definition only. The class name ALL is reserved.

event-name
Specifies the name of an event to add to the class. To define a class that contains multiple events, specify a separate CLASS option for each event.

Qualifier

/ITEMS=(item-name[, . . .])
/NOITEMS (default)
Specifies a set of items (or item groups) to collect for an event when the selection specifies this class. You can supply an asterisk (*) as a wildcard symbol to collect all of the items specified in the EVENT /ITEMS qualifier. The default is /NOITEMS.

DEFAULT_CLASS

The DEFAULT_CLASS option designates which class to collect when the facility selection does not specify a class. If you omit the DEFAULT_CLASS option, the ALL class is designated as the default class. By default, DECtrace places each event in the facility class named ALL.

Format

DEFAULT_CLASS class-name

Parameter

class-name
Specifies the name of the default class. It must be the class ALL or a previously defined class. Digital recommends that the class most frequently referred to by facility selections be specified as the default class.

7 DECtrace Commands

DECtrace provides a command-line interface. To use the DECtrace commands, preface them with the keyword COLLECT. For example:

```
$  COLLECT SHOW VERSION
DECtrace Version V1.0-0
$
```

For better user interface performance, you can enter the DECtrace command environment by entering the COLLECT command with no arguments. This eliminates binding to the history and administration databases for each command. DECtrace prompts you for commands until you return to DCL command level with the EXIT command. For example:

```
$  COLLECT
DECtrace>  SHOW VERSION
DECtrace Version V1.0-0
DECtrace>  EXIT
$
```

This chapter describes the format and usage of each DECtrace command. Table 7–1 summarizes the available commands.

Table 7–1 DECtrace Commands

Command	Description
@ (Execute Procedure)	Executes the commands in a command file as if you had typed them at the DECtrace> prompt.
CANCEL COLLECTION	Stops data collection for an active collection. If a collection is pending, it removes the collection from the schedule.
CREATE DEFINITION[1]	Creates a facility definition in the DECtrace administration database.
CREATE SELECTION	Creates a facility selection in the DECtrace administration database. Selection names must be unique in the DECtrace administration database. If you use the /REPLACE qualifier, your new facility selection replaces the old selection of the same name.
DELETE DEFINITION	Deletes a facility definition from the DECtrace administration database.
DELETE SELECTION	Deletes a facility selection from the DECtrace administration database.
EXIT	Exits from DECtrace and returns to the DCL command level.
EXTRACT DEFINITION	Extracts a facility definition from the DECtrace administration database and stores it in a binary file.
FORMAT	Formats one or more DECtrace data files into a formatted data file or database.
HELP	Displays requested information about the DECtrace commands.
INSERT DEFINITION	Inserts a facility definition in a binary format into the DECtrace administration database.
REPORT[2]	Generates a report based on formatted data from one or more collections.
SCHEDULE COLLECTION	Schedules data collection based on the specified qualifiers.

[1]Following the CREATE DEFINITION command section are separate reference sections on these facility definition options: CLASS, DEFAULT_CLASS, EVENT, GROUP, and ITEM.

[2]Following the REPORT command section are separate reference sections on these report options: EVENT, ITEM, and RESTRICTION.

(continued on next page)

Table 7–1 (Cont.) DECtrace Commands

Command	Description
SET HISTORY	Changes the history database that DECtrace uses for the SHOW HISTORY command or creates a new history database.
SHOW COLLECTION	Shows data collection information in one of two formats: BRIEF or FULL.
SHOW DEFINITION	Shows information about one or more facility definitions registered in the DECtrace administration database in one of two formats: FULL or NAMES_ONLY.
SHOW HISTORY	Shows all error and/or informational messages that have occurred during one or all data collections active on your system.
SHOW REGISTER	Shows the individual processes for which data can be collected.
SHOW SELECTION	Displays a facility selection in one of three formats: BRIEF, FULL, or NAMES_ONLY.
SHOW VERSION	Shows the version number of DECtrace installed on your system.
SPAWN	Creates a subprocess of the current process.
STOP SYSTEM	Stops DECtrace and interrupts any active data collection.

@ (Execute Procedure)

The at sign (@) means execute, just as in DCL. When you type @ and the name of an indirect command file, DECtrace executes the statements in that file as if you had typed them one at a time at the DECtrace> prompt. The command file must be a VMS text file that contains DECtrace commands.

Format

@ file-spec

Parameter

file-spec
The name of the indirect command file. You can specify a full VMS file specification, a file name, or a logical name. If you specify a file name, DECtrace looks in your current default VMS directory for a file by that name. The file must contain valid DECtrace commands. The default file type is COM.

Description

This command is useful because it allows you to prepare a sequence of commands that you use often and that must be repeated identically each time you issue them. For example, if you generate weekly or monthly reports, you will want the same parameters defined each time.

Example

```
DECtrace> @DISK$USER1:[SMITH.COMS]SCHEDULE_A_COLLECTION.COM
```

Executes the commands in the file SCHEDULE_A_COLLECTION.COM, where the command file contains the following lines:

```
CREATE SELECTION VAX_INFO /OPTIONS -
  FACILITY RDBVMS
  FACILITY ACMS/CLASS=ALL
  EXIT
SCHEDULE COLLECTION MY_TEST MY_DATA.DAT -
  /SELECTION=VAX_INFO -
  /BEGIN=11:00 /END=12:00
EXIT
```

CANCEL COLLECTION

Stops one or more active or pending collections.

Format

CANCEL COLLECTION [collection-name]

Command Qualifiers	**Defaults**
/[NO]CONFIRM	/CONFIRM
/SELECTION=selection-name	/SELECTION=*

Parameter

collection-name
The name of the data collection that you want to cancel. The collection name is a unique text string that identifies one currently scheduled collection. You can use an asterisk (*) as a wildcard to cancel all collections.

The collection name is optional; however, you must specify either the collection name or a facility selection name (with the /SELECTION qualifier). If you specify both a collection name and a selection name, the facility selection must have been specified in the original SCHEDULE COLLECTION command or DECtrace issues an error message and does not cancel any data collection.

Qualifiers

/CONFIRM (default)
/NOCONFIRM
Specifies whether DECtrace prompts you to confirm the cancellation of each collection (similar to the VMS DCL command DELETE/CONFIRM).

If you use the /CONFIRM qualifier, DECtrace identifies each collection that matches the specified collection name and asks if you want to cancel it. If you use /NOCONFIRM, DECtrace cancels the specified collection(s) without prompting for confirmation.

/SELECTION=selection-name

Specifies a facility selection that has one or more collections active or pending on the current system. If you specify an asterisk (*) as a wildcard character in place of the selection name, DECtrace cancels all data collection.

The /SELECTION qualifier is optional; however, you must specify either a facility selection name or a collection name. If not, DECtrace issues an error message indicating that the command is ambiguous.

If you specify both a facility selection name and a collection name, the selection must have been specified in the original SCHEDULE COLLECTION command or DECtrace issues an error message and does not cancel any data collection.

Description

In a cluster, the CANCEL COLLECTION command cancels data collection either locally or cluster-wide, depending on how the collection was originally scheduled. If you scheduled data collection with the /NOCLUSTER qualifier, CANCEL COLLECTION cancels the collection locally. If you scheduled data collection with the /CLUSTER qualifier, CANCEL COLLECTION cancels the collection on all nodes in the cluster.

If both a collection name and a facility selection name are supplied, they must be the same as those given in the original SCHEDULE COLLECTION command.

To cancel data collection, you must be the originator of the collection, or have VMS BYPASS or SYSPRV privilege.

Examples

1

```
$ COLLECT CANCEL COLLECTION MONDAYS_TEST /CONFIRM
Cancel MONDAYS_TEST [N]: YES
%EPC-S-SCHED_CANCEL, Data collection MONDAYS_TEST is cancelled
```

Cancels the data collection named MONDAYS_TEST.

2

```
$ COLLECT CANCEL COLLECTION /SELECTION=ACMS_DATA /NOCONFIRM
%EPC-S-SCHED_CANCEL, Data collection MY_TEST is cancelled
%EPC-S-SCHED_CANCEL, Data collection TUESDAYS_TEST is cancelled
%EPC-S-SCHED_CANCEL, Data collection WEDNESDAYS_TEST is cancelled
```

Cancels all data collection associated with the facility selection ACMS_DATA. In this example, three collections are canceled.

3

```
$ COLLECT CANCEL COLLECTION MY_TEST /SELECTION=JOES_DATA /NOCONFIRM
%EPC-E-SCHED_XFAILED, Cancel collection operation failed
%EPC-E-SCHED_NTFST, No collections found matching collection MY_TEST
 in selection JOES_DATA
```

Attempts to cancel the data collection named MY_TEST. However, MY_TEST was not scheduled with the facility selection JOES_DATA, so DECtrace issues an error message and does not cancel the collection.

CREATE DEFINITION

Creates a facility definition and defines its events.

Format

CREATE DEFINITION facility-name facility-id

Command Qualifiers	**Defaults**
/EVENTS=(event-name[, . . .])	See text
/[NO]OPTIONS[=file-spec]	/NOOPTIONS
/[NO]REPLACE	/NOREPLACE
/VERSION="version-code"	See text

Parameters

facility-name
The name of the facility that you want to define. The maximum length of the facility name is 27 characters.

facility-id
A unique numeric identifier for the facility. Facility IDs in the range 1 to 2047 are reserved to Digital. You can define facilities with IDs in the range 2048 to 4095. Note that the facility ID must match the ID specified in the EPC$INIT service routine call (in the program source code).

All versions of a given facility use the same facility ID.

Qualifiers

/EVENTS=(event-name[, . . .])
Specifies the events that the facility includes. If you use the /EVENTS qualifier to define events, the events collect data on the resource utilization items only, and you cannot specify additional items.

The /EVENTS and /OPTIONS qualifiers are mutually exclusive, but you must have one or the other.

/OPTIONS[=file_spec]
/NOOPTIONS (default)
Specifies a file containing facility definition options. If you do not include a file specification, the Option> prompt is displayed and you can enter your options interactively. Facility options are order-dependent. Each option must be defined before it is referenced.

The /EVENTS and /OPTIONS qualifiers are mutually exclusive, but you must have one or the other.

Following the CREATE DEFINITION command are individual descriptions of the facility definition options.

/REPLACE
/NOREPLACE (default)
Specifies that the current facility definition replaces any previously existing facility definition with the same facility name and version code. If a facility definition exists and you attempt to redefine it without using the /REPLACE qualifier, DECtrace issues an error message and does not store the new definition.

If no facility definition with the same name and version code exists, DECtrace ignores the /REPLACE qualifier.

/VERSION="version-code"
Specifies the version of the facility. The name of the version can be any printable string up to 10 characters. You must enclose the text string with quotation marks (" "). This is a required qualifier and must match the version specified in the EPC$INIT service routine call.

Description

The CREATE DEFINITION command creates a facility definition that is stored in the DECtrace administration database, which is available cluster-wide. The definition is retained until you explicitly delete it using the DELETE DEFINITION command. DECtrace facility definitions consist of:

- Class definitions (optional)
- Creation date
- Creator's username
- Event definitions (optional)
- Event names

- Facility ID
- Facility name
- Item definitions (optional)
- Item group definitions (optional)
- Version code

Use the CREATE DEFINITION command to define the facility name, ID, version code, and event names. DECtrace automatically records your username and the creation date as part of the facility definition.

You need VMS SYSPRV or BYPASS privilege to create a registered facility definition.

Examples

1

```
$ COLLECT CREATE DEFINITION DATA_ENTRY 2052 /EVENTS=(INIT,STOP) -
_$ /VERSION="V1.0"
```

Creates the facility definition for version V1.0 of the DATA_ENTRY facility and specifies that the default resource utilization items should be collected for the events INIT and STOP. The facility ID for DATA_ENTRY is 2052.

2

```
$ COLLECT CREATE DEFINITION MY_APPLICATION 2050 -
_$ /VERSION="T1.0-3" /OPTIONS=MY_APP_OPTIONS.TXT
```

Creates the facility definition for version T1.0-3 of the MY_APPLICATION facility and specifies that the facility definition options listed in the file MY_APP_OPTIONS.TXT be used. The facility ID for MY_APPLICATION is 2050.

3

```
$ COLLECT CREATE DEFINITION /OPTIONS NEW_TOOL 2049 /VERSION="T1.0-1"
Option> ITEM IMAGE_NAME TEXT/SIZE=256
Option> EVENT INVOCATION/ITEMS=IMAGE_NAME
Option> CTRL/Z
$
```

Creates a facility definition for version T1.0-1 of the NEW_TOOL facility. The facility has one event (INVOCATION) which has one item (IMAGE_NAME) associated with it. The facility ID for NEW_TOOL is 2049.

CREATE DEFINITION Options—ITEM

The ITEM option identifies and describes the characteristics of a data item that the facility can collect.

Format

ITEM item-name datatype

Command Qualifiers	**Defaults**
/CHARACTERISTICS=(characteristic[, . . .])	/CHARACTERISTICS=PRINTABLE
/IDENTIFIER=item-id	Next unused item-id
/RADIX=base-system	/RADIX=DECIMAL
/REPORT_HEADER="text"	/REPORT_HEADER=item-name
/REPORT_WIDTH=number-of-spaces	See text
/SIZE=n-bytes	See text
/USAGE_TYPE=usage-type	None

Parameters

item-name
Specifies a text string that names the item. The string must be a valid VMS name, cannot end with a dollar sign ($) or underscore (_), and cannot be an Rdb/VMS reserved word.

datatype
Specifies the data type of the item. Table 7–2 lists the valid data types.

Table 7–2 ITEM Data Types

Data Type	Description
ASCIC	Varying-length counted string of up to 255 bytes
ASCIW	Varying-length counted string of up to 16,383 bytes
BYTE	Signed byte
FIXED_ASCIC	Fixed-length counted string of up to 255 bytes

(continued on next page)

Table 7–2 (Cont.) ITEM Data Types

Data Type	Description
LONGWORD	Signed longword (4 bytes)
QUADWORD	Signed quadword (8 bytes)
WORD	Signed word (2 bytes)

Qualifiers

/CHARACTERISTICS=(characteristic[, . . .])
Either PRINTABLE or NONPRINTABLE that specifies whether the contents of an item can be displayed. The default is PRINTABLE.

/IDENTIFIER=item-id
An integer between 1 and 100 that specifies a unique identifier for the item. (Identifiers 101 to 128 are reserved for DECtrace-defined items.) The default is the next unused ID between 1 and 100.

/RADIX=base-system
Either HEXADECIMAL or DECIMAL that specifies the base of the number system used for numerical items. The default radix is DECIMAL.

/REPORT_HEADER="text"
A text string that specifies a heading to display when creating reports using this item. The default is the name of the item.

/REPORT_WIDTH=number-of-spaces
An integer that specifies the width of the column to use when displaying item values in a report. The default width depends on the item data type, as listed in Table 7–3.

Table 7–3 ITEM Default Report Widths

Data Type	Width
ASCIC	80 columns
ASCIW	80
BYTE	4
FIXED_ASCIC	80
LONGWORD	11
QUADWORD	32
WORD	6

/SIZE=n-bytes

An integer that specifies the maximum size in bytes of the ASCIC, FIXED_ ASCIC, and ASCIW data types. The /SIZE qualifier is required for these data types and ignored for all others.

/USAGE_TYPE=usage-type

One of the keywords in Table 7–4 that specifies how the data is to be used.

Table 7–4 ITEM Usage Types

Usage Type	Description
COUNTER	Typically a running count or total; its value either always increases or always decreases for the duration of the event. In a DECtrace Summary Report, for items with usage type COUNTER, DECtrace displays the difference between the start and end values.
LEVEL	A meter or gauge that indicates the current value of some metric; its value may increase or decrease over the duration of the event.
PERCENT	A percentage; similar to LEVEL, except that it has upper and lower limits of 100 and 0, respectively.
PRIVATE	Facility-defined data that does not fall into one of the other usages. DECtrace does not attempt to display this item on DECtrace reports.
TEXT	Text characters.

Valid usage types depend on the data type of the item, as shown in Table 7–5.

Table 7–5 ITEM Usage Types by Data Type

Data Type	Valid Usage Types	Default
ASCIC	TEXT, PRIVATE	TEXT
ASCIW	TEXT, PRIVATE	TEXT
BYTE	COUNTER, LEVEL, PERCENT, PRIVATE	COUNTER
FIXED_ASCIC	TEXT, PRIVATE	TEXT
LONGWORD	COUNTER, LEVEL, PERCENT, PRIVATE	COUNTER
QUADWORD	COUNTER, LEVEL, PERCENT, PRIVATE	COUNTER
WORD	COUNTER, LEVEL, PERCENT, PRIVATE	COUNTER

CREATE DEFINITION Options—GROUP

The GROUP option allows you to refer to a set of items by a single name. You can use a group name in the /ITEMS qualifier of an EVENT or CLASS option to refer to all of the items within that group.

Format

GROUP group-name

Command Qualifier	**Default**
/ITEMS=(item-name[, . . .])	None

Parameter

group-name
Specifies the name of the group.

Qualifier

/ITEMS=(item-name[, . . .])
Specifies the items contained in the group. This is a required qualifier.

CREATE DEFINITION Options—EVENT

The EVENT option identifies and describes the characteristics of an event including the set of items to collect and (optionally) the items to collect when the event is a point event, start event, or end event. By default, DECtrace places each event in the facility class named ALL.

Format

EVENT event-name

Command Qualifiers	Defaults
/END_EVENT=(item-name[, . . .])	See text
/FIRST_SEGMENT_SIZE=n-bytes	None
/IDENTIFIER=event-id	Next unused event ID
/[NO]ITEMS=(item-name[, . . .])	/NOITEMS
/POINT_EVENT=(item-name[, . . .])	See text
/REPORT_HEADER=text	/REPORT_HEADER=event-name
/SEGMENT_SIZE=n-bytes	None
/START_EVENT=(item-name[, . . .])	See text

Parameter

event-name
Specifies the name of the event. The event name must be a valid VMS name and cannot end with a dollar sign ($) or underscore (_).

Qualifiers

/END_EVENT=(item-name[, . . .])
Specifies the items that DECtrace collects at the end of a duration event. If you omit the /END_EVENT qualifier, DECtrace uses the /ITEMS qualifier to define the items to collect at the end of a duration event.

/FIRST_SEGMENT_SIZE=n-bytes
Specifies the size of the first string segment for the event relation in the formatted database. If you do not specify a size for the first segment, the DECtrace formatting component uses 64 bytes.

See the OPTIMIZATION PARAMETERS under the FORMAT command and Section A.1.2 for more information about string segmentation.

/IDENTIFIER=item-id

Specifies an integer between 1 and 128 that uniquely identifies the event. The default is the next unused event identifier between 1 and 128.

/ITEMS=(item-name[, . . .])
/NOITEMS (default)

Defines the set of all items to collect for an event. The default is /NOITEMS. This is a required qualifier.

/POINT_EVENT=(item-name[, . . .])

Specifies the items that DECtrace collects for a point event. If you omit the /POINT_EVENT qualifier, DECtrace uses the /ITEMS qualifier to define the items to collect for a point event.

/REPORT_HEADER=text

Specifies the text to display as a heading when creating reports using this event. The default report heading is the name of the event.

/SEGMENT_SIZE=n-bytes

Specifies the size of the remaining string segments (the first having been specified with the /FIRST_SEGMENT_SIZE qualifier) for the event relation in the formatted database. If you do not specify a size for the segments, the DECtrace formatting component uses 128 bytes.

See the OPTIMIZATION PARAMETERS under the FORMAT command and Section A.1.2 for more information about string segmentation.

/START_EVENT=(item-name[, . . .])

Specifies the items that DECtrace collects at the beginning of a duration event. If you omit the /START_EVENT qualifier, DECtrace uses the /ITEMS qualifier to define the items to collect at the beginning of a duration event.

CREATE DEFINITION Options—EVENT

Description

In the /ITEMS, /START_EVENT, /END_EVENT, and /POINT_EVENT, qualifiers, items are collected in the order specified. You can specify:

- Any item or item group that has already been defined.
- A predefined resource utilization item.
- The predefined item group RESOURCE_ITEMS, which collects all resource utilization items.
- An asterisk (*) as a wildcard symbol, meaning all of the ITEM options in the facility definition as well as resource utilization items. (This is for the /ITEMS qualifier only.)

CREATE DEFINITION Options—CLASS

The CLASS option defines a new class if the specified class does not already exist and binds an event (with a subset of its associated items) to a class.

Format

CLASS class-name event-name

Command Qualifier	**Default**
/[NO]ITEMS=(item-name[, . . .])	/NOITEMS

Parameters

class-name
Specifies the name of the class. A class name must be a unique 1- to 32-character string consisting of alphanumeric characters, dollar signs ($), and underscores (_) . The string must be unique within the context of the facility definition. The class name ALL is reserved.

event-name
Specifies the name of an event to add to the class. To define a class that contains multiple events, specify a separate CLASS option for each event.

Qualifier

/ITEMS=(item-name[, . . .])
/NOITEMS (default)
Specifies a set of items or item groups to collect for an event when the selection specifies this class. You can supply an asterisk (*) as a wildcard symbol to collect all of the items specified in the EVENT /ITEMS qualifier. The default is /NOITEMS.

CREATE DEFINITION Options—DEFAULT_CLASS

The DEFAULT_CLASS option designates which class to collect when the facility selection does not specify a class. If you omit the DEFAULT_CLASS option, the ALL class is designated as the default class. By default, DECtrace places each event in the facility class named ALL.

Format

DEFAULT_CLASS class-name

Parameter

class-name
Specifies the name of the default class. It must be the class ALL or a previously defined class. Digital recommends that the class most frequently referred to by facility selections be specified as the default class.

CREATE SELECTION

Creates a facility selection, that is, a list of the facilities for which to collect data. You can optionally specify the class of data to collect for each facility.

Format

CREATE SELECTION selection-name

Command Qualifiers	Defaults
/[NO]COMMENT="comment-string"	/NOCOMMENT
/FACILITY=(facility-name[, . . .])	/FACILITY=*
/[NO]OPTIONS[=file-spec]	/NOOPTIONS
/[NO]REPLACE	/NOREPLACE

Parameter

selection-name
Specifies the name of the facility selection that you want to define. The selection name must be a unique 1- to 32-character string consisting of alphanumeric characters, dollar signs ($), and underscores (_). The selection name must be unique for the local system or for the entire cluster in a cluster environment.

Qualifiers

/COMMENT
/NOCOMMENT (default)
Allows you to include a comment string describing the selection. The comment can be up to 80 characters and must be enclosed with quotation marks (" ").

/FACILITY=(facility-name[, . . .])
Specifies one or more facilities for which to collect data. You can use an asterisk (*) as a wildcard to have DECtrace collect data for all available facilities.

Use the /FACILITY qualifier to define simple facility selections where you want to collect data from the default class for one or more facilities. If you want to specify a collection class for individual facilities, use the /OPTIONS qualifier. The /FACILITY and /OPTIONS qualifiers are mutually exclusive.

If you do not specify either the /FACILITY or the /OPTIONS qualifier, the facility selection describes the default class data for all facilities.

/OPTIONS[=file-spec]
Specifies a file that contains the details of the facility selection. The default file type for the options file is OPT.

If you enter a VMS file specification with the /OPTIONS qualifier, DECtrace uses that file as a source for facility selection options. If you enter /OPTIONS without a file specification, DECtrace prompts you for the options interactively. Press CTRL/Z or type EXIT to exit from the Option> prompt.

The /FACILITY and /OPTIONS qualifiers are mutually exclusive. If you do not specify either the /FACILITY or the /OPTIONS qualifier, the facility selection describes the default class data for all facilities. See the Description section for information about facility selection options.

/REPLACE
/NOREPLACE (default)
Specifies that the current facility selection replaces any previously existing selection with the same name. If a facility selection exists and you attempt to redefine it without using the /REPLACE qualifier, DECtrace issues a warning message and does not store the new selection.

If no facility selection with the same name exists in the DECtrace administration database, the /REPLACE qualifier is ignored.

Description

DECtrace facility selections consist of:

- The name of the selection
- A list of facilities for which to collect data
- A collection class for each facility

You use the CREATE SELECTION command to choose a subset of the available facilities for which you want to collect data. You can define most general-purpose selections with the command without the /OPTIONS qualifier. For example, the following command defines the facility selection ACMS_DATA to collect data for the default class for VAX ACMS:

```
$ COLLECT CREATE SELECTION ACMS_DATA /FACILITY=ACMS
```

However, to define a more detailed facility selection, you should use an options file. The /OPTIONS qualifier allows you to specify a different collection class for each facility from which you want to collect data. The qualifier takes a file name as an argument. The options file lists each facility and the collection class you want to use. Each facility is described on a separate line in the file. If you specify /OPTIONS but do not include a file name, DECtrace prompts you for the options. The format of the facility description in the options file is:

```
FACILITY facility-name [/VERSION="version-code"] [/CLASS=class-name]
```

facility-name
The name of the facility for which to collect data.

version-code
A text string identifying the version of the facility. The string must be enclosed with quotation marks (" ").

class-name
The class of data that you want collected for the facility. Facilities registered with DECtrace can have one or more collection classes associated with them. These classes are subsets of the available events and items chosen for their significance to a specific function. For example, a facility might have specific collection classes defined for performance, workload, or capacity planning. You use the SHOW DEFINITION /FORMAT=NAMES_ONLY command to determine what classes are available for a given facility. All facilities have the ALL class which contains all of the events and associated items for the facility. Furthermore, one of the classes is designated as the default class based on its predicted importance and frequency of use.

Examples

1

```
$ COLLECT CREATE SELECTION ACMS_AND_RDB /FACILITY=(ACMS,RDBVMS)
```

Creates a facility selection that collects the default events for both the ACMS and Rdb/VMS facilities. The selection name is ACMS_AND_RDB.

2

```
$ COLLECT CREATE SELECTION VAX_INFO_DATA /OPTIONS
Option> FACILITY RDBVMS
Option> FACILITY ACMS/CLASS=ALL
Option> FACILITY DBMS/VERSION="V4.1"/CLASS=WORKLOAD
Option> CTRL/Z
```

Creates a facility selection that describes the following data:

- Default collection class data for the latest version of the Rdb/VMS facility
- All data for the ACMS facility
- Workload class data for Version 4.1 of the DBMS facility

3

```
$ COLLECT CREATE SELECTION TP_DATA /OPTIONS=TP_FACS.OPT
```

Creates a facility selection that collects data on the facilities listed in the options file TP_FACS.DAT.

DELETE DEFINITION

Deletes one or more facility definitions from the DECtrace administration database. Use this command when you install a new version of a product and want to remove the facility definition for the previous version. Deleting registered facility definitions (those in the range of 1 to 2047) is a management function that requires VMS BYPASS or SYSPRV privilege to perform. For non-registered facilities, you must have created the definition, or have VMS BYPASS or SYSPRV privilege.

Format

DELETE DEFINITION facility-name [, . . .]

Command Qualifiers	**Defaults**
/[NO]CONFIRM	/CONFIRM
/VERSION="version-code"	See text

Parameter

facility-name
Specifies the name of one or more facilities for which you want to delete the facility definition. You cannot use any wildcard characters.

Qualifiers

/CONFIRM (default)
/NOCONFIRM
Specifies whether DECtrace prompts you to confirm the deletion of each facility definition (similar to the VMS DCL command DELETE/CONFIRM).

If you use the /CONFIRM qualifier, DECtrace identifies each facility definition and version code that matches the specified facility name and version code and asks if you want to delete it. If you use /NOCONFIRM, DECtrace deletes the specified facility definition from the DECtrace administration database without prompting for confirmation.

/VERSION="version-code"
Specifies the version of the facility for which you want to delete the facility definition. You must enclose the text string with quotation marks (" "). If there is more than one version of the facility on the system, you can use the

/VERSION qualifier to specify which version you want to delete. However, you can use an asterisk (*) as a wildcard character to delete the facility definitions for all versions of a given facility.

This is a required qualifier.

Description

Use the DELETE DEFINITION command to delete facility definitions from the DECtrace administration database. You do not delete the actual facility images from your system, but merely the DECtrace facility definition for that facility.

You cannot delete a facility definition if any active or pending collections are collecting data from that facility. Use the SHOW SELECTION /FORMAT=BRIEF command to determine if a facility has active or pending data collection associated with it. If it does, you must use the CANCEL COLLECTION command to stop data collection before you can delete the facility definition. Also, any facility selections that specify the facility definition explicitly must be deleted. If a facility selection does not specify the version of the facility and another version exists on the system, you can delete the facility definition without canceling scheduled data collection.

Examples

1

```
$ COLLECT DELETE DEFINITION TRANS2 /NOCONFIRM /VERSION="V2.0"
%EPC-S-FACDEL_DELETED, Facility definition TRANS2 V2.0 was deleted
```

Deletes the TRANS2 version V2.0 facility definition from the DECtrace administration database.

2

```
$ COLLECT DELETE DEFINITION TRANS2 /VERSION="V2.0"
Delete TRANS2 V2.0 [N]: YES
%EPC-S-FACDEL_DELETED, Facility definition TRANS2 V2.0 was deleted
```

Deletes the TRANS2 facility definition from the DECtrace administration database after prompting for confirmation.

3

```
$ COLLECT DELETE DEFINITION  MY_APPL /NOCONFIRM /VERSION=*
%EPC-S-FACDEL_DELETED, Facility definition MY_APPL V4.2 was deleted
%EPC-S-FACDEL_DELETED, Facility definition MY_APPL V4.3 was deleted
%EPC-S-FACDEL_DELETED, Facility definition MY_APPL V4.4 was deleted
```

Deletes the facility definitions for all versions of the facility MY_APPL by using the /VERSION qualifier.

DELETE SELECTION

Deletes one or more facility selections from the DECtrace administration database.

Format

DELETE SELECTION selection-name [, . . .]

Command Qualifier	**Default**
/[NO]CONFIRM	/CONFIRM

Parameter

selection-name
Specifies the name of one or more facility selections that you want to delete. You cannot use wildcard characters.

Qualifier

/CONFIRM (default)
/NOCONFIRM
Specifies whether DECtrace prompts you to confirm the deletion of each facility selection (similar to the VMS DCL command DELETE/CONFIRM).

If you use the /CONFIRM qualifier, DECtrace identifies each selection name that matches the name you specified and asks if you want to delete it. If you use /NOCONFIRM, DECtrace deletes all matching facility selections without prompting for confirmation.

Description

You can delete a facility selection from the DECtrace administration database with the DELETE SELECTION command. You must be the creator of the selection or have VMS BYPASS or SYSPRV privilege. You cannot delete a facility selection if any data collection, either active or pending, is using that facility selection.

DELETE SELECTION

To delete a facility selection that is being used in any current or scheduled data collections, you must first cancel the data collection. See Section 3.4 for information on how to cancel all data collection using a particular facility selection.

Examples

1
```
$ COLLECT DELETE SELECTION ACMS_DATA /NOCONFIRM
%EPC-S-SELDEL_DELETED, Selection ACMS_DATA was deleted)
```

Deletes the ACMS_DATA facility selection from the DECtrace administration database.

2
```
$ COLLECT DELETE SELECTION RDB_DATA
Delete RDB_DATA [N]: YES
%EPC-S-SELDEL_DELETED, Selection RDB_DATA was deleted)
```

Deletes the RDB_DATA facility selection from the DECtrace administration database after prompting you for confirmation.

EXIT

Exits DECtrace and returns to the DCL command level. Also used to exit from the Option> prompt and return to the DECtrace> prompt.

Format

EXIT

Description

Use the EXIT command to quit the DECtrace prompting mode after entering a series of commands. Alternately, you can press CTRL/Z at the DECtrace> prompt.

Examples

1

```
DECtrace> EXIT
$
```

Exits DECtrace and returns to DCL command level.

2

```
DECtrace> CTRL/Z
$
```

Exits DECtrace and returns to DCL command level.

3

```
DECtrace> CREATE SELECTION INFO_DATA /OPTIONS
Option> FACILITY ACMS/CLASS=ALL
Option> FACILITY DBMS/VERSION="V4.1"/CLASS=WORKLOAD
Option> EXIT
DECtrace>
```

Exits from the DECtrace options environment and returns to the DECtrace> prompt.

EXTRACT DEFINITION

Extracts a facility definition from the DECtrace administration database and stores it in a binary file. You can later use the INSERT DEFINITION command to store the extracted facility definition in another DECtrace administration database.

Format

EXTRACT DEFINITION facility-name file-spec

Command Qualifier	Default
/VERSION="version-code"	See text

Parameters

facility-name
Specifies the name of the facility definition that you want to extract into a binary file. You cannot use wildcard characters.

file-spec
Specifies the name of the file that DECtrace creates to store the binary form of the facility definition. The default device and directory are your VMS default device and directory. The default file type is EPC$DEF.

Qualifier

/VERSION="version-code"
Specifies the version of the facility. The name of the version can be any printable text string up to 10 characters. You cannot use wildcard characters, and the string must be enclosed with quotation marks (" ").

This is a required qualifier.

Description

Use the EXTRACT DEFINITION command to copy facility definitions from one system to another. The binary file contains the entire facility definition, including the original creation date. This allows an exact replica of the facility definition to be moved to a remote site and included in the remote site's DECtrace administration database by using the INSERT DEFINITION command.

You need VMS BYPASS, READALL, or SYSPRV privilege to extract either a registered facility definition or the definition for a facility created by another user.

Example

```
$ COLLECT EXTRACT DEFINITION MY_FACILITY MY_FAC031 /VERSION="V3.1"
%EPC-S-FACEXT, Facility definition(s) was successfully extracted
```

Creates a file containing the facility definition for version V3.1 of MY_FACILITY. DECtrace uses your default device and directory and the default file type.

FORMAT

Formats one or more DECtrace data files into one formatted data file or database.

Format

FORMAT data-file[, . . .] [formatted-file-or-database]

Command Qualifiers	**Defaults**
/[NO]CDDPLUS_DEFINITIONS=pathname	/NOCDDPLUS_DEFINITIONS
/[NO]FILELIST	/NOFILELIST
/[NO]MERGE	/NOMERGE
/RDBVMS_OPTIMIZATION=(param=value[, . . .])	See text
/TYPE=output-type	/TYPE=RDBVMS

Parameters

data-file

Must be either of the following:

- The name of a data capture file produced by data collection. You can also specify data capture files from more than one collection, provided all the collections reference the same facility selection.
- The name of a file that contains a list of data collection file names produced by one or more collections. (See the description of the /FILELIST qualifier.)

If the data collection files contain data from collections that are associated with different facility selections, DECtrace issues an error and no formatting is done.

You cannot use any wildcard characters in the data file specifications. The default device and directory are your default VMS device and directory. The default file type is DAT.

formatted-file-or-database

Specifies the name of the formatted data file or database that DECtrace creates to store the data. The default device and directory are your VMS default device and directory. The /TYPE qualifier determines whether DECtrace creates a VAX RMS file or an Rdb/VMS database.

For VAX RMS formatted files, the default file type is DAT. For Rdb/VMS databases, you *cannot* specify a file type because Rdb/VMS creates both a database and a snapshot file. If you do specify a file type and request an Rdb/VMS database, DECtrace returns an error message and does not format the file.

The formats of both the VAX RMS file and the Rdb/VMS database are described in Appendix A.

Note *DECtrace reports can be generated only from a formatted Rdb / VMS database.*

Qualifiers

/CDDPLUS_DEFINITIONS=pathname
/NOCDDPLUS_DEFINITIONS (default)
Specifies whether to create VAX CDD/Plus record definitions for the formatted database or data file records. If you request VAX CDD/Plus record definitions, you must specify a pathname where the definitions will be stored in the VAX CDD/Plus dictionary.

You can only specify the /CDDPLUS_DEFINITIONS qualifier when you are formatting your data collection file(s) into an Rdb/VMS formatted database.

/FILELIST
/NOFILELIST (default)
Specifies that the first parameter to the FORMAT command is a **file list**, which is a file containing a list of file specifications for the data files you want to format. For example:

```
$  COLLECT FORMAT MY_LIST.TXT /FILELIST MY_DATA
```

Each file specification must be on a separate line within the file list. For example, MY_LIST.TXT contains the following lines:

```
DISK$USER1:[SMITH.COLLECTOR]DATA1.DAT
DISK$USER2:[DATA]DATA2.DAT
DISK$USER3:[DATA]DATA3.DAT
```

The default file type for the file list is TXT.

/MERGE
/NOMERGE (default)
Specifies that the data in the data files should be merged with existing data in the specified formatted output file or database. The /MERGE qualifier lets you combine data from previous collections (that have already been formatted)

with data from new collections. Only data from collections scheduled using the same facility selection can be combined.

When you specify the /MERGE qualifier and the output file or database does not exist, DECtrace issues an error and does not perform any formatting. If you do not specify /MERGE, and the output file already exists, DECtrace creates a new version of the file using the same name.

/RDBVMS_OPTIMIZATION=(param=value[, . . .])
Specifies to use optimization parameters when creating an Rdb/VMS formatted database. Table 7–6 shows the parameters and their default. Each parameter is described in the following section under **Optimization Parameters**.

Table 7–6 Rdb/VMS Optimization Parameters

Optimization Parameter	Default Value
ALLOCATION	2000 database pages
BUFFER_SIZE	30 blocks
[NO]JOURNAL	JOURNAL
MIN_EXTENT	500 pages
NUM_BUFFERS	30 buffers
[NO]STRING_OPTIMIZATION	STRING_OPTIMIZATION
[NO]VIEWS	VIEWS

/TYPE=output-type
Specifies the format of the formatted data file. Valid format types are:

RDBVMS
RMS

If you use the RDBVMS keyword, DECtrace creates an Rdb/VMS database that you can use with the REPORT command to generate reports. If you use the RMS keyword, DECtrace creates a VAX RMS sequential file. The choice of output types lets you select the format best suited for your purposes. With either type, you can choose to write your own reports by interpreting the data through user-written reporting programs or a report generator such as VAX DATATRIEVE.

RDBVMS is the default output type. The /TYPE=RMS qualifier is provided for users who plan to create their own reports based on the collected data. The formats of the both the VAX RMS file and the Rdb/VMS database are described in Appendix A.

Optimization Parameters

ALLOCATION=number-of-pages
Specifies the number of database pages allocated for the data file produced when you create an Rdb/VMS database. You may want to increase the value of this qualifier when creating a very large Rdb/VMS database from your data collection file(s). The recommended value for the ALLOCATION parameter is based on the following formula:

$$\textit{number-of-pages} = (4000 + \textit{size-of-DCF})/2$$

Where *size-of-DCF* is the size in blocks of all data collection files.

The default allocation is 2000 pages.

BUFFER_SIZE=number-of-blocks
Specifies the number of blocks allocated per buffer for use in creating an Rdb/VMS database. Valid buffer sizes range from 1 to 64.

The BUFFER_SIZE and NUM_BUFFERS parameters enable you to reduce the direct I/O required during formatting operations. However, the direct I/O is inversely proportional to the page faulting required; as you reduce the direct I/O, you increase your process's page faulting because the I/O operations have been transferred to the paging disk. In general, you can safely increase the number of buffers and the buffer size only if your system has a large amount of memory. You should experiment with different values to determine the optimal size and number of buffers for your application.

The default buffer size is 30 blocks.

[NO]JOURNAL
Specifies whether the database should be recoverable in the event of a formatting failure. If you specify NOJOURNAL, the database will be corrupted if formatting of a file fails. The disadvantage of using JOURNAL, is that formatting operations take longer to complete.

Note that if you specify NOJOURNAL when formatting a group of data collection files, and one of the files fails to format correctly, the database will be corrupted from that point on in the formatting. In this case, the format operation may fail completely.

The default is JOURNAL.

MIN_EXTENT=number-of-pages
Specifies the minimum number of pages used for the data file extent. You may want to increase the value of this qualifier when creating a very large Rdb/VMS database from your data collection file(s).

The default minimum extent is 500 pages.

NUM_BUFFERS=number-of-buffers
Specifies the number of buffers allocated for the creation of an Rdb/VMS database. The number is an an unsigned integer greater than zero.

The BUFFER_SIZE and NUM_BUFFERS parameters enable you to reduce the direct I/O required during formatting operations. However, the direct I/O is inversely proportional to the page faulting required; as you reduce the direct I/O, you will increase your process's page faulting because the I/O operations have been transferred to the paging disk. In general, you can safely increase the number of buffers and the buffer size only if your system has a large amount of memory. You should experiment with different values to determine the optimal size and number of buffers for your application.

The default number of buffers is 30.

[NO]STRING_OPTIMIZATION
Specifies whether string storage optimization should be performed using a segmentation storage scheme. This parameter can greatly reduce the size of the formatted database for data of large ASCIC, ASCIW, and FIXED_ASCIC types. However, it will be more difficult to write report generators for a database using this optimization. DECtrace reporting is unaffected by the optimization.

The STRING_OPTIMIZATION parameter takes a list of the following parameters:

- FIRST_SEGMENT_SIZE=n-bytes—the size of the first string segment stored in the event data relation.
- SEGMENT_SIZE=n-bytes—the size of additional segments stored in the segmented string relation.

The default is STRING_OPTIMIZATION and the default values of the optimization parameters are based on the data specified in the facility definition.

[NO]VIEWS
Specifies whether views should be created for the event-data relations. If so, DECtrace creates four views for each event-data relation.

For data collection files with a large number of different event types, you can greatly increase formatting performance by specifying the NOVIEWS parameter. Note that DECtrace reporting does not need views.

The default is VIEWS.

Description

DECtrace collects raw data from the facilities that you select. The raw data must then be formatted into either an Rdb/VMS database or a VAX RMS file in order to be used. To use DECtrace reporting, you must specify /TYPE=RDBVMS (or use the default to the /TYPE qualifier) to create an Rdb/VMS database. For example, the following command creates a formatted Rdb/VMS database named ALL_MY_DATA.RDB:

```
$  COLLECT FORMAT DUA0:[DATA]FOO1.DAT,DUA1:[DATA]FOO2.DAT ALL_MY_DATA
```

Examples

1

```
$ COLLECT FORMAT MY_COLLECTION.DAT TEST_DATA
```

Formats the data in the file MY_COLLECTION.DAT and stores the formatted data in an Rdb/VMS database in your current default VMS device and directory under the name TEST_DATA.RDB.

2

```
$ COLLECT FORMAT MY_LIST.TXT /FILELIST MY_DATA
```

Formats the data in the files listed in MY_LIST.TXT and stores the formatted data in an Rdb/VMS database in your current default VMS device and directory under the name MY_DATA.RDB. MY_LIST.TXT contains a list of data file names, one per line.

3

```
$ COLLECT FORMAT WEEK3.DAT JUNE_DATA /MERGE
```

Formats the data in WEEK3.DAT and merges it with the data already in the previously formatted JUNE_DATA.RDB database.

4

```
$ COLLECT FORMAT ONE_BIG_FILE.DAT ALL_DATA -
_$ /RDBVMS_OPTIMIZATION=(NUM_BUFFERS=40, BUFFER_SIZE=50, NOVIEWS, -
_$ STRING_OPTIMIZATION=(FIRST=32,SEGMENT=64), ALLOCATION=10000)
```

Formats the data in ONE_BIG_FILE.DAT and stores the formatted data in an Rdb/VMS database named ALL_DATA.RDB. Because the file is very large, the normal formatting parameters are increased to provide more optimal performance.

HELP

Displays information about DECtrace and DECtrace commands.

Format

HELP [topic] . . .

Parameter

topic
Specifies the subject about which you want information from the HELP library. You can specify the names of one main topic and up to eight subtopics with the HELP command.

Example

```
DECtrace> HELP CREATE
  CREATE

    Creates a facility definition or a facility selection.

  Additional information available:

  DEFINITION     SELECTION

CREATE Subtopic?
```

Invokes DECtrace HELP to display information about the CREATE DEFINITION and CREATE SELECTION commands and prompts for further information.

INSERT DEFINITION

Inserts a facility definition in a binary format (previously created by the EXTRACT DEFINITION command) into the DECtrace administration database.

Format

INSERT DEFINITION file-spec

Command Qualifiers	**Defaults**
/[NO]LIBRARY	/NOLIBRARY
/[NO]REPLACE	/NOREPLACE

Parameter

file-spec
Specifies the name of the binary file that contains the facility definition. The default device and directory are your VMS default device and directory. The default file type is EPC$DEF.

Qualifiers

/LIBRARY
/NOLIBRARY (default)
Inserts the facility definition into the DECtrace facility library (EPC$FACILITY.TLB) located in SYS$COMMON:[SYSLIB]. This text library is referenced during reinstallations of DECtrace. If you are going to remove DECtrace from your system but wish to retain all of your facility definitions, you need to extract the definitions from the DECtrace administration database (with the EXTRACT DEFINITION command) and re-insert them into the DECtrace facility library.

/REPLACE
/NOREPLACE (default)
Specifies that the facility definition replaces any previously existing facility definition with the same facility name and version code. If a facility definition exists and you attempt to insert a matching facility and version without using the /REPLACE qualifier, DECtrace issues a warning and does not store the new definition.

If no facility definition with the same name and version code exists, the /REPLACE qualifier is ignored.

Description

The EXTRACT DEFINITION and INSERT DEFINITION commands allow you to replicate facility definitions on multiple systems. This is useful when you want to test an application in several different environments.

You need VMS SYSPRV or BYPASS privilege to replace either a registered facility definition or the definition for a facility created by another user.

Examples

1

```
$ COLLECT INSERT DEFINITION MY_FAC031 /REPLACE
```

Inserts the facility definition in the binary file MY_FAC031.EPC$DEF into the DECtrace administration database.

2

```
$ COLLECT INSERT DEFINITION NEW_TOOL /LIBRARY
```

Inserts the facility definition in the binary file NEW_TOOL.EPC$DEF into the DECtrace administration database and the DECtrace facility library.

REPORT

Generates a report based on formatted data from one or more collections.

Format

REPORT formatted-database

Command Qualifiers	**Defaults**
/BEFORE="time"	All data in file
/EVENTS=(event-name[, . . .])	/EVENTS=*
/FACILITY=(facility-name[, . . .])	/FACILITY=*
/INTERVAL=time-unit	/INTERVAL=MINUTES
/LENGTH=number-of-lines	/LENGTH=60
/[NO]OPTIONS=file-spec	/NOOPTIONS
/OUTPUT=file-spec	/OUTPUT=SYS$OUTPUT
/SINCE="time"	All data in file
/STATISTICS=(stat-type[, . . .])	/STATISTICS=COUNT
/TITLE="text-string"	/TITLE=""
/TYPE=report-type	/TYPE=SUMMARY
/WIDTH=number-of-columns	/WIDTH=80

Parameter

formatted-database
Specifies the name of a formatted database created with the FORMAT command. If the specified parameter is not an Rdb/VMS database, DECtrace issues an error message and generates a 0-block output file.

This is a required parameter.

Qualifiers

/BEFORE="dd-mmm-yyyy hh:mm:ss"
Specifies that DECtrace reports on all data collected at or before the indicated time. By default, DECtrace reports on all data in the formatted database.

If you use the /BEFORE qualifier, you must specify the time as an absolute date and time using a quoted string. You cannot use a VMS delta time.

/EVENTS=(event-name[, . . .])
Specifies one or more events to report on. By default, DECtrace reports on the data collected for all facilities and events contained in the formatted database. You can use the /EVENTS qualifier to restrict the report to specific events.

If you specify the /EVENTS qualifier, you must list the facilities you want with the /FACILITY qualifier. You cannot use the default /FACILITY=* if you use the /EVENTS qualifier. An error occurs if you specify a facility which does not contain the specified event(s).

The /EVENTS and /OPTIONS qualifiers are mutually exclusive.

/FACILITY=(facility-name[, . . .])
Specifies one or more facilities to report on. By default, DECtrace reports on the data collected for all facilities and events contained in the formatted database. You can use the /FACILITY qualifier to restrict the report to specific facilities.

If you specify the /FACILITY qualifier, you do not have to use the /EVENTS qualifier. If you do, the effects are cumulative: for example, if you specify two facilities and one event, DECtrace reports on that event for both facilities, provided both facilities contain the event. Note that an error occurs if you specify a facility which does not contain the specified event(s).

The /FACILITY and /OPTIONS qualifiers are mutually exclusive.

/INTERVAL=time-unit
Specifies the time interval for FREQUENCY reports. FREQUENCY reports calculate the number of occurrences of an event or events at given time intervals during data collection. The /INTERVAL qualifier specifies the time interval at which event occurrences are calculated. The time interval must be one second, one minute, or one hour, specified as one of the following keywords:

SECONDS
MINUTES
HOURS

The default interval is one minute.

The /INTERVAL qualifier is valid for FREQUENCY reports only. DECtrace ignores the qualifier for other types of reports. Use the /TYPE qualifier to specify a FREQUENCY report.

/LENGTH=number-of-lines
Specifies the length of each page of the report in number of lines. The default length is 60 lines.

/OPTIONS[=file-spec]
/NOOPTIONS (default)
Specifies an option file that describes the contents and format of the report. The default file type for the options file is OPT.

If you supply a VMS file specification with the /OPTIONS qualifier, DECtrace uses that file as a source for the report characteristics. If you enter /OPTIONS without a file specification, DECtrace prompts you for the options. See Section 4.8 for a description of the report characteristics that you can specify in an options file.

The /EVENTS and /OPTIONS qualifiers are mutually exclusive.

/OUTPUT=file-spec
Specifies the destination for the report. By default, DECtrace displays the report on the SYS$OUTPUT device (usually, your terminal). The /OUTPUT qualifier can redirect the output to a file or another device.

/SINCE="dd-mmm-yyyy hh:mm:ss"
Specifies that DECtrace reports on all data collected at or after the specified time. By default, DECtrace reports on all the collected data in the formatted database.

If you use the /SINCE qualifier, you must specify the time as an absolute date and time using a quoted string. You cannot use a VMS delta time.

/STATISTICS=(stat-type[, . . .])
Specifies the statistics to include for each data item in a SUMMARY report. Valid statistics are:

ALL
COUNT (default)
MAXIMUM
MEAN
MINIMUM
STANDARD_DEVIATION
TOTAL
95_PERCENTILE

The /STATISTICS qualifier is valid for SUMMARY reports only. DECtrace ignores the qualifier for DETAIL and FREQUENCY reports.

/TITLE="text-string"
Specifies the title of the report. The title is displayed at the top of each page of the report. The title is a text string that can contain any printable characters. The string must be enclosed with quotation marks (" ").

/TYPE=report-type
Specifies the type of report to create. The valid report types are:

DETAIL
FREQUENCY
SUMMARY

The default report type is SUMMARY. See Section 4.4 for a detailed description of each report.

/WIDTH=number-of-columns
Specifies the width of each page of the report in number of characters. Valid widths are 80 and 132. If you specify a value other than 80 or 132, or if you do not specify a width, DECtrace uses 80 characters.

Description

In the simplest case, the REPORT command creates a summary report, displayed to SYS$OUTPUT, on all of the data in the formatted database. For example, the following command generates a report based on the data in the formatted database MY_TEST.RDB:

```
$  COLLECT REPORT MY_TEST.RDB
```

You can further refine the report with the /FACILITY and /EVENTS qualifiers. For example, the following command reports on only the ACMS PROCEDURE_CALL events:

```
$  COLLECT REPORT MY_TEST -
_$  /FACILITY=ACMS /EVENT=PROCEDURE_CALL
```

The /BEFORE, /INTERVAL, /SINCE, and /STATISTICS qualifiers provide additional customizing features that you may want to use.

You can specify all of the characteristics of the report using the REPORT qualifiers listed in the previous paragraph. However, the qualifiers are global; that is, the statistics that you request with the /STATISTICS qualifier apply to all of the events that you ask to be included in the report. You can override the qualifiers for each event. If you want to define separate characteristics for each event, you can use the /OPTIONS qualifier.

See Section 4.8 for an example of the report characteristics that you can specify in the report options file. Also, following the REPORT command in this reference section are descriptions of the report options EVENT, ITEM, and RESTRICTION.

Examples

1

```
$ COLLECT REPORT MY_REPORT /OUTPUT=MY_REPORT.TXT
```

Creates a summary report on all of the data in the formatted file MY_REPORT.RDB and stores the report in the file MY_REPORT.TXT.

2

```
$ COLLECT REPORT MY_REPORT /OUTPUT=MY_REPORT.TXT -
_$ /FACILITY=ACMS /EVENT=TASK /STATISTICS=(MINIMUM,MAXIMUM,MEAN)
```

Creates a summary report of ACMS task data stored in the formatted database MY_REPORT.RDB. The report lists the minimum, maximum, and mean values of task data items. DECtrace stores the report in the file MY_REPORT.TXT.

3

```
$ COLLECT REPORT MY_REPORT /OUTPUT=MY_REPORT.TXT /OPTIONS
Option> EVENT TASK /FACILITY=ACMS /STATISTICS=MAXIMUM -
_Option> /GROUP_BY=TASK_NAME
Option> CTRL/Z
```

Creates a summary report of ACMS task data from the formatted database MY_REPORT.RDB. The report lists the maximum number of occurrences of the TASK event, grouped by task name. DECtrace stores the report in the file MY_REPORT.TXT.

REPORT Options—EVENT

Defines the characteristics of a report on a specific event. You specify this command using the /OPTIONS qualifier to the REPORT command.

Format

EVENT event-name

Command Qualifiers	**Defaults**
/BEFORE=time	All data in file
/FACILITY=facility-name	None
/GROUP_BY=(item-name[, . . .])	See text
/INTERVAL=time	See text
/[NO]ITEMS=(item-name[, . . .])	/ITEMS=*
/SINCE=time	All data in file
/STATISTICS=(stat-type[, . . .])	See text
/[NO]SUBTITLE="text-string"	/NOSUBTITLE
/TYPE=report-type	See text

Parameter

event-name
Specifies the name of an event to report on.

Qualifiers

/BEFORE="dd-mmm-yyyy hh:mm:ss"
Specifies that DECtrace reports on all data collected at or before to the indicated time. By default, DECtrace reports on all data in the formatted database for this event.

If you use the /BEFORE qualifier, you must specify the time as an absolute date and time using a quoted string. You cannot use VMS delta time.

/FACILITY=facility-name
Specifies the name of the facility that contains the event.

/GROUP_BY=(item-name[, . . .])

Specifies the item or items by which to group event occurrences. In SUMMARY and FREQUENCY reports, the report statistics are divided into groups based on equivalent values of the specified items. The final group displays grand total statistics for the entire report.

In DETAIL reports, event occurrences are listed in ascending order based on the item values. If the event is a duration event, the end value of the item is used, providing the item was collected on the end event. Otherwise, the start value is used.

For segmented ASCII strings, the /GROUP_BY qualifier is based on the first segment in the event relation. If two items are not equal, but the difference does not occur until after the first segment, the report considers them equal.

/INTERVAL=time-unit

Specifies the time interval for FREQUENCY reports. FREQUENCY reports calculate the number of occurrences of an event or events at given time intervals during data collection. The /INTERVAL qualifier specifies the time interval at which event occurrences are calculated. The time interval must be one second, one minute, or one hour specified as one of the following keywords:

SECONDS
MINUTES
HOURS

If an /INTERVAL qualifier is specified on the REPORT command line, the command line qualifier specifies the default interval. The /INTERVAL qualifier on an EVENT option overrides the default interval for this event only.

The default interval is one minute.

The /INTERVAL qualifier is valid for FREQUENCY reports only. The qualifier is ignored for other types of reports.

/ITEMS=(item-name[, . . .])
/NOITEMS

Specifies the item or items on which to report. The /ITEMS qualifier is valid for DETAIL and SUMMARY reports only. It is ignored for FREQUENCY reports.

If you specify /NOITEMS and /GROUP_BY on a DETAIL report, DECtrace ignores the /GROUP_BY qualifier. The resulting report contains only the report header information.

If you specify /NOITEMS on a SUMMARY report, DECtrace uses /STATISTICS=COUNT and ignores any other statistic that you specify in the command.

/SINCE="dd-mmm-yyyy hh:mm:ss"
Specifies that DECtrace reports on all data collected for this event at or after the specified time. By default, DECtrace reports on all the collected data in the formatted database for this event name.

If you use the /SINCE qualifier, you must specify the time as an absolute date and time using a quoted string. You cannot use a VMS delta time.

/STATISTICS=(stat-type[, . . .])
Specifies the summary statistics to include for each item. Valid statistics are:

95_PERCENTILE
ALL
COUNT (default)
MAXIMUM
MEAN
MINIMUM
STANDARD_DEVIATION
TOTAL

If you specify the /STATISTICS qualifier on the REPORT command line, the command line qualifier specifies the default statistics. The /STATISTICS qualifier on an EVENT option overrides the default statistics for this event only.

The /STATISTICS qualifier is valid for SUMMARY reports only. DECtrace ignores the qualifier for DETAIL and FREQUENCY reports.

/SUBTITLE="text-string"
/NOSUBTITLE (default)
Specifies a subtitle for the report. The subtitle is displayed only at the top of the first page of the report. The subtitle is a text string that can contain any printable characters. The string must be enclosed with quotation marks (" ")

/TYPE=report-type

Specifies the type of report to create. The valid report types are:

DETAIL
FREQUENCY
SUMMARY

If you specify the /TYPE qualifier on the REPORT command line, the command line qualifier specifies the default report type. The /TYPE qualifier on an EVENT option overrides the default report type for this event only.

The default report type is SUMMARY.

Description

Specify options either in an options file or at the Option> prompt. In either case, you must use the /OPTIONS qualifier to the REPORT command to specify options. Specify each option on a separate line. You can specify each option more than once.

REPORT Options—ITEM

Defines the display characteristics of items in the report. You specify this command using the /OPTIONS qualifier to the REPORT command.

Display characteristics include:

- Report heading
- Display width

The default display characteristics of data items are defined in the facility definition. The ITEM report option allows you to customize these characteristics for an item pertaining to the preceding EVENT report option. The characteristics defined in an ITEM option remain in effect only for a single EVENT report option.

Format

ITEM item-name

Command Qualifiers	**Defaults**
/REPORT_HEADER=text	See text
/WIDTH=number-of-columns	See text

Parameter

item-name
The name of the item for which to define the characteristics.

Qualifiers

/REPORT_HEADER=text
Specifies the text to display as a heading in the report when listing data for the specified item.

The facility definition specifies the default report heading.

/WIDTH=number-of-columns
Specifies the number of columns to reserve for displaying data for the specified item.

The facility definition specifies the default width.

REPORT Options—RESTRICTION

Defines a subset of the data to report on for the preceding EVENT report option. You specify this command using the /OPTIONS qualifier to the REPORT command.

You can selectively report on data based on the following items:

- Collection names
- Image names
- Item values
- Node names
- Process IDs (EPID)

Format

```
                ( COLLECTION collection-name [, . . . ] )
                | EPID process-ID [, . . . ]            |
RESTRICTION    <  IMAGE image-name [, . . . ]            >
                | ITEM item-name item-value             |
                ( NODE node-name [, . . . ]             )
```

Parameters

collection-name
The name of one or more collections to use to select data for reporting. Only data collected during the specified collections is reported.

image-name
The name of the executable image which registered with the DECtrace Registrar process. Only data collected from processes running the specified image(s) is reported.

The image name that you specify is used as a wildcard in searching through the formatted database for the desired images. The effect is that the restriction is done on **image-name**.

item-name
The name of an item to use to select data for reporting.

item-value
The value of the item to use to select data. Only records that match the item name and value are reported.

node-name
The name of one or more network nodes to use to select data for reporting. Only data collected on the specified nodes is reported.

process-ID
The EPIDs of one or more processes to use to select data for reporting. Only data collected from the specified processes is reported.

SCHEDULE COLLECTION

Schedules data collection to occur on the cluster or local node.

Format

SCHEDULE COLLECTION collection-name data-collection-file[, . . .]

Command Qualifiers	**Defaults**
/BEGINNING=time	/BEGINNING=current-time
/[NO]CLUSTER	/CLUSTER
/DURATION=time	See text
/ENDING=time	See text
/[NO]FILELIST	/NOFILELIST
/PROTECTION=(ownership:access[, . . .])	See text
/REGISTRATION_ID=(registration-id[, . . .])	/REGISTRATION_ID=*
/SELECTION=selection-name	See text

Parameters

collection-name

The name of the collection you are scheduling. The collection name must be a unique 1- to 32-character string consisting of alphanumeric characters, dollar signs ($), and underscores (_). The collection name must be unique for the local system or for the entire cluster in a cluster environment.

data-collection-file

The name of the file or files to create for storing the collected data.

The file specification can consist of a device, directory, file name, and file type. The device and directory default to your current VMS default device and directory.

DECtrace creates the data collection file(s) when the collection is scheduled, not when data collection actually begins. The default protection on the data collection file is your default protection. You can set the protection on the file using the /PROTECTION qualifier. For example, by removing world write access and specifying group write access, you can restrict data collection to only those processes that are in your UIC group.

You can specify more than one data file to spread the output over multiple disks and avoid I/O bottlenecks during collection. See the FORMAT command for instructions on combining these multiple data files afterwards.

Qualifiers

/BEGINNING=time

Specifies when data collection starts. You can specify the starting time using either a VMS absolute or delta time format, including both date and time.

By default, data collection starts immediately. If you specify a relative date and time, the starting time is determined by the relative offset from the current time. For example, if you ask for "+1:" at 10:00, the test begins at 11:00.

The starting time for data collection must be in the future. If you specify a starting time that has already passed, DECtrace issues an error message and does not schedule data collection.

/CLUSTER (default)
/NOCLUSTER

Specifies whether data collection occurs only on the local node or on all nodes of the cluster.

By default, SCHEDULE COLLECTION schedules data collection cluster-wide (that is, data collection occurs on every node in the cluster). You use the /NOCLUSTER qualifier to schedule data collection on only the current node.

On a standalone system, the /CLUSTER qualifier is ignored.

/DURATION=time

Specifies the length of time over which data collection occurs. You must specify the duration as a VMS delta time.

The /DURATION and /ENDING qualifiers are mutually exclusive.

/ENDING=time

Specifies when data collection ends. You can specify the ending time using either a relative or absolute VMS time format, including both date and time.

You must specify an ending time or use the /DURATION qualifier. If you do not include the /ENDING or /DURATION qualifiers, DECtrace issues an error message and does not schedule data collection.

If you specify a relative date and time, the ending time is determined by the relative offset from the current time, *not* the starting time.

If the ending time is earlier than the starting time, DECtrace issues an error message and does not schedule data collection.

/FILELIST
/NOFILELIST (default)
Specifies that the second parameter to the SCHEDULE COLLECTION command is a **file list**, which is a file containing a list of file specifications to use as output data files for data collection. One advantage of using a file list is that you can specify multiple data files without worrying about the DCL limitations on the length of a command line.

The /FILELIST qualifier is position-dependent; you must specify it on the second parameter to the command. For example:

```
$ COLLECT SCHEDULE COLLECTION MY_TEST DATA_FILE.TXT /FILELIST
```

Each file specification must be on a separate line within the file list. For example:

```
DISK$USER1:[SMITH.COLLECTOR]DATA1.DAT
DISK$USER2:[DATA]DATA2.DAT
DISK$USER3:[DATA]DATA3.DAT
```

The default file type for the file list is TXT.

/PROTECTION=(ownership:access[, . . .])
Specifies the protection on the data collection file(s). A process must have write access to a file in order to record event data in the file.

Valid ownership categories are (S)ystem, (O)wner, (G)roup, and (W)orld.

A valid access code is any combination of the following: (R)ead, (W)rite, (E)xecute, and (D)elete.

If you do not specify a value for each ownership category, or if you omit the /PROTECTION qualifier, DECtrace applies the current default protection for each unspecified category. If the data collection file replaces a previous version, the protection scheme of the old file is used on the new one.

/REGISTRATION_ID=(registration-id[, . . .])
Identifies facility- and process-specific registration IDs to limit the number of processes collecting data. By default, no restrictions are placed on data collection.

When an image instrumented with DECtrace activates, several process-specific registration IDs are registered automatically. By specifying the appropriate combinations of the process ID (EPID), process name, image name, and user name, you can collect data on a per-process, per-image, or per-user basis. Use the SHOW REGISTER command to display the registration information associated with a process.

A facility can pass a facility-specific registration ID on its EPC$INIT call. This allows you to specify a subset of the processes using a facility. For example, the ACMS facility can restrict the processes collecting data to those that use specific applications. In this case, the registration ID is the application name. Note that not all facilities choose to provide a registration ID.

If you specify a registration ID, it must be exactly as it appears in the SHOW REGISTER display. You cannot use abbreviations or default values.

See Section 3.1.2 for more information about registration IDs.

/SELECTION=selection-name
Specifies the facility selection to use for the collection. The facility selection lists the facilities for which data will be collected and specifies a collection class for each.

This is a required qualifier.

Description

You can schedule data collection cluster-wide or on a subset of the nodes in a VAXcluster. To schedule data collection on a subset of a cluster, you must log in to each node on which you want to schedule the collection and start local data collection on each of those nodes.

Only one collection can be active on a particular node at any time. You can schedule many different collections on your local system or cluster, but the collections cannot have overlapping time intervals. However, you can schedule overlapping collections on different subsets of your cluster. For example, you could schedule a collection on two nodes and have a different collection active on the remaining nodes in the cluster at the same time.

The situations that result in an error are:

- If the ending time is earlier than the starting time
- If the starting date and time are in the past
- If another collection is active or is scheduled to become active on that node during the same time interval

See Section 3.2 for more information on scheduling data collection.

Examples

1

```
$ COLLECT SCHEDULE COLLECTION MY_TEST MY_DATA.DAT -
_$ /SELECTION=ACMS_DATA -
_$ /BEGIN=11:00 /END=12:00 -
_$ /NOCLUSTER
%EPC-S-SCHED, Data collection MY_TEST is scheduled
```

Schedules the collection MY_TEST to run on the local node and to begin at 11:00 and end at 12:00 on the current day. DECtrace stores the collected data in the file MY_DATA.DAT in your default device and directory and uses your default protection scheme.

2

```
$ COLLECT SCHEDULE COLLECTION WEDNESDAYS_TEST WEDNESDAY_DATA.DAT -
_$ /SELECTION=RDB_AND_ACMS -
_$ /BEGINNING="+1-" /DURATION="2:" -
_$ /PROTECTION=(W:W) -
_$ /CLUSTER
%EPC-S-SCHED, Data collection WEDNESDAYS_TEST is scheduled
```

Schedules a cluster-wide collection to begin at the current time on the following day. The name of the collection is WEDNESDAYS_TEST. It collects the data described by the RDB_AND_ACMS facility selection. Data is collected for two hours and is stored in the file WEDNESDAY_DATA.DAT.

3

```
$ COLLECT SCHEDULE COLLECTION ALL_DAY LOTSOFDATA.DAT -
_$ /SELECTION=JUST_RDB -
_$ /DURATION="2:" -
_$ /CLUSTER -
_$ /PROTECTION=(G:W) -
_$ /REGISTRATION_ID=(DISK1:[PAYROLL.IMAGES]WEEKLY_CHECKS.EXE)
```

Schedule data collection to begin immediately and run for two hours. The collection runs on the entire cluster and collects Rdb/VMS event data from the WEEKLY_CHECKS program.

SET HISTORY

Temporarily changes the history database that DECtrace uses for the SHOW HISTORY command, or creates a new history database.

Format

SET HISTORY [history-file]

Command Qualifier	**Default**
/[NO]NEW_FILE	/NONEW_FILE

Parameter

history-file
Specifies the history database to use when displaying history information with the SHOW HISTORY command. The SET HISTORY command remains in effect for the life of the user's process or until another SET HISTORY command is issued. The default file type is RDB.

If you do not specify the /NEW_FILE qualifier, then the history-file is a required parameter.

Qualifier

/NEW_FILE
/NONEW_FILE (default)
Specifies that a new history database will be created. The /NEW_FILE qualifier does not accept a file specification. DECtrace creates the new history file with the logical name EPC$HISTORY_DB and the next higher version number.

Use of the /NEW_FILE qualifier requires VMS OPER privilege.

Description

The SET HISTORY command performs two separate functions:

- Setting an alternate history file for SHOW HISTORY commands
- Creating a new history file for recording DECtrace messages

If you specify the history-file parameter, DECtrace points all SHOW HISTORY commands to the specified file. This command is useful for examining the information in an old history file. Note that the scope of this command is limited to within the DECtrace> prompt context for this process. If you exit from the DECtrace command environment and then re-enter it, all subsequent SHOW HISTORY commands will reference the current, active history file.

If you specify the /NEW_FILE qualifier, DECtrace creates a new history file and writes any new messages to the new file. A slight lapse between the time you issue the SET HISTORY command and the time DECtrace starts writing to the new file should not exceed 30 seconds. In the meantime, history records continue to be written to the old file.

If you specify both a history file name and the /NEW_FILE qualifier, DECtrace issues an error message and does not perform either function.

Examples

1

```
$ COLLECT
DECtrace> SET HISTORY OLD_HISTORY
DECtrace> SHOW HISTORY
```

Temporarily changes the history file for SHOW HISTORY commands to the file OLD_HISTORY.RDB so that you can display information from an old history database.

2

```
$ COLLECT SET HISTORY /NEW_FILE
```

Creates a new history file with the same file name as referred to by the logical name EPC$HISTORY_DB and a version number one higher than the current file.

SHOW COLLECTION

Displays information about active or pending data collection on your system.

Format

SHOW COLLECTION [collection-name]

Command Qualifiers	**Defaults**
/[NO]CLUSTER	/CLUSTER
/FORMAT=type	/FORMAT=BRIEF
/OUTPUT=file-spec	/OUTPUT=SYS$OUTPUT
/SELECTION=selection-name	/SELECTION=*

Parameter

collection-name

Specifies the name of the data collection for which to display information. You can omit the collection name or use an asterisk (*) as a wildcard character to display information about all collections.

If you specify both a facility selection (with the /SELECTION qualifier) and a collection, DECtrace uses the collection name first. If the data collection does not use the specified selection, DECtrace executes the command using the collection name but also issues a warning message.

Qualifiers

/CLUSTER (default)
/NOCLUSTER

Specifies whether to display information about data collection scheduled only on the local node, or about data collection scheduled on any node in the cluster.

By default, SHOW COLLECTION displays information about data collection scheduled on any node in the cluster.

On a standalone system, the /CLUSTER qualifier is ignored.

/FORMAT=type

Specifies the amount of information to display. Valid types are:

BRIEF
FULL

The default format is BRIEF, which displays the collection name, facility selection name, and starting and stopping times for data collection. If you specify /FORMAT=FULL, DECtrace displays a complete description of the data collection.

/OUTPUT=file-spec

Specifies the destination for the display. By default, DECtrace displays the information on the current SYS$OUTPUT device (usually, your terminal). The /OUTPUT qualifier can redirect the output to a file or another device.

/SELECTION=selection-name

Specifies the name of a facility selection for which you want to see schedule information. If you use an asterisk (*) as a wildcard character, or if you omit both the collection name and the /SELECTION qualifier, DECtrace displays information about all data collection currently running or scheduled for the system or VAXcluster.

Description

See Section 3.5.1 for a full description of the information in the SHOW COLLECTION displays.

Examples

1
```
$ COLLECT SHOW COLLECTION /SELECTION=ACMS_DATA /NOCLUSTER
```

Displays information about any data collection running or scheduled to run on the local node that uses the facility selection ACMS_DATA.

2
```
$ COLLECT SHOW COLLECTION WEDNESDAYS_TEST
```

Displays information about the data collection WEDNESDAYS_TEST.

SHOW DEFINITION

Displays information about one or more facility definitions registered in the DECtrace administration database.

Format

SHOW DEFINITION [facility-name]

Command Qualifiers	**Defaults**
/FORMAT=type	/FORMAT=NAMES_ONLY
/OUTPUT=file-spec	/OUTPUT=SYS$OUTPUT
/VERSION="version-code"	/VERSION=*

Parameter

facility-name
Specifies the name of the facility definition you want information about. You can use an asterisk (*) as a wildcard character to display information about all of the facilities defined on the system.

If you do not specify a facility name, DECtrace displays information on all the currently defined facilities.

Qualifiers

/FORMAT=type
Specifies the amount of information to display. Valid types are:

FULL
NAMES_ONLY

The default format type is NAMES_ONLY.

/OUTPUT=file-spec
Specifies the destination for the display. By default, DECtrace displays the information on the current SYS$OUTPUT device (usually, your terminal). The /OUTPUT qualifier can redirect the output to a file or another device.

/VERSION="version-code"
Specifies the version of the facility that you want information about. You must enclose the text string with quotation marks (" ").

If you do not specify a version code or if you use an asterisk (*) as a wildcard character, DECtrace displays information about all versions of the facility that you name.

Description

See Section 6.4 for a description of each type of SHOW DEFINITION display.

Examples

1
```
$ COLLECT SHOW DEFINITION /FORMAT=NAMES_ONLY
23-DEC-1989 11:42       Facility Definition Information       Page 1
Names Only Report                                          DECtrace V1.0

 Facility:           Version:    Creation Date:      Class:
 ------------------  ----------  ------------------  ------------
 ATM_SAMPLE          V1.0        25-AUG-1989 14:17   ALL          (D)
 MY_FACILITY         V4.2        23-OCT-1989 17:09   ALL          (D)
```

Displays the names, versions, creation dates, and collection classes for all of the facility definitions on the system.

2
```
$ COLLECT SHOW DEFINITION MY_FACILITY /VERSION="V4.2" /FORMAT=FULL
23-Dec-1989 11:43      Facility Definition Information        Page 1
Full Report                                                DECtrace V1.0

Facility:        MY_FACILITY
Number:          2048
Version:         V4.2
Creation date:   13-Oct-1989 17:09
Created by:      SMITH

Events:
   .
   .
   .
Items:
   .
   .
   .
```

Displays complete information about the MY_FACILITY Version V4.2 facility definition.

SHOW HISTORY

Displays information about the DECtrace system, including:

- When the system started and stopped
- When collections started, stopped, or were canceled
- When processes registered and unregistered
- Any errors that occurred in the DECtrace registration and collecting processes

Format

SHOW HISTORY [collection-name]

Command Qualifiers	Defaults
/BEFORE=time	All data in file
/[NO]CHRONOLOGICAL	/NOCHRONOLOGICAL
/[NO]CLUSTER	/CLUSTER
/FORMAT=type	/FORMAT=ERROR
/NODE="node-name"	/NODE=*
/OUTPUT=file-spec	/OUTPUT=SYS$OUTPUT
/SINCE=time	All data in file

Parameter

collection-name
The name of the collection for which to display information. By default, DECtrace displays the history for all collections. You can display information for a single collection by specifying a collection name on the command line. You cannot use wildcard characters.

Qualifiers

/BEFORE=time
Specifies that only information before a specific date and time be displayed. By default, DECtrace displays all information in the history file.

/CHRONOLOGICAL
/NOCHRONOLOGICAL (default)
Specifies for history messages to be displayed in chronological order with headers displayed for each message. This qualifier is used primarily for debugging purposes in a cluster environment.

/CLUSTER (default)
/NOCLUSTER
Specifies whether DECtrace displays information on the current node only or on all nodes in the VAXcluster.

By default, SHOW HISTORY displays cluster-wide information.

On a standalone system, the /CLUSTER qualifier is ignored.

/FORMAT=type
Specifies the type of messages that should be displayed. By default, SHOW HISTORY displays only the error messages. The valid types are:

ALL
INFORMATIONAL
ERROR (default)

/NODE="node-name"
Selects only those messages from the specified node. By default, DECtrace displays history messages from all nodes in the VAXcluster.

/OUTPUT=file-spec
Specifies the destination for the display. By default, DECtrace displays the information on the current SYS$OUTPUT device (usually, your terminal). The /OUTPUT qualifier can redirect the output to a file or another device.

/SINCE=time
Specifies that only information after a specific date and time be displayed. By default, DECtrace displays all of the information in the history file.

Description

The SHOW HISTORY command can be used in conjunction with the SET HISTORY command to display information from older history databases.

See Section 3.5.2 for a description of the information in the SHOW HISTORY displays.

Examples

1
```
$ COLLECT SHOW HISTORY
```

Displays the errors which have occurred for the DECtrace system on the local VAXcluster (in a cluster environment).

2
```
$ COLLECT SHOW HISTORY /FORMAT=INFORMATIONAL /OUTPUT=HISTORY.LOG
```

Writes a description of all informational messages and events to the file HISTORY.LOG.

3
```
$ COLLECT
DECtrace> SET HISTORY OLD_HISTORY
DECtrace> SHOW HISTORY
```

Temporarily changes the history file for SHOW HISTORY commands to the file OLD_HISTORY.RDB so that you can display information from an old history database.

4
```
$ COLLECT SHOW HISTORY /NODE=MYVAX2 /FORMAT=ALL
```

Displays all of the errors and messages for node MYVAX2.

SHOW REGISTER

Displays information about processes currently registered with the DECtrace administration database, the facilities and registration IDs for the processes, and whether the processes are currently collecting data.

Format

SHOW REGISTER

Command Qualifiers	Defaults
/[NO]CLUSTER	/CLUSTER
/OUTPUT=file-spec	/OUTPUT=SYS$OUTPUT

Qualifiers

/CLUSTER
/NOCLUSTER (default)
Specifies whether DECtrace displays information on the processes on the local node only or on all nodes in the VAXcluster.

On a standalone system, the /CLUSTER qualifier is ignored.

/OUTPUT=file-spec
Specifies the destination for the display. By default, DECtrace displays the information on the current SYS$OUTPUT device (usually, your terminal). The /OUTPUT qualifier can redirect the output to a file or another device.

Description

See Section 3.1.2 for a description of the information in the SHOW REGISTER display.

Example

```
$ COLLECT SHOW REGISTER
```

Displays the names and process IDs of processes registered with the DECtrace administration database.

SHOW SELECTION

Displays information about one or more facility selections stored in the DECtrace administration database on the current system or VAXcluster.

Format

SHOW SELECTION [selection-name]

Command Qualifiers	**Defaults**
/FORMAT=type	/FORMAT=BRIEF
/OUTPUT=file-spec	/OUTPUT=SYS$OUTPUT

Parameter

selection-name
The name of the facility selection for which you want information. You can use an asterisk (*) as a wildcard character to display information about all facility selections defined on the system.

If you do not specify a selection name, DECtrace displays information on all of the facility selections defined on the system.

Qualifiers

/FORMAT=type
Specifies the amount of information to display. Valid types are:

BRIEF
FULL
NAMES_ONLY

The default format is BRIEF.

/OUTPUT=file-spec
Specifies the destination for the display. By default, DECtrace displays the information on the current SYS$OUTPUT device (usually, your terminal). The /OUTPUT qualifier can redirect the output to a file or another device.

Description

See Section 2.4 for a full description of the information in the SHOW SELECTION displays.

Examples

1
```
$ COLLECT SHOW SELECTION /FORMAT=NAMES_ONLY
```

Displays the names of all facility selections on the system.

2
```
$ COLLECT SHOW SELECTION ACMS_DATA /FORMAT=FULL -
_$ /OUTPUT=ACMS_DATA_FULL.LIST
```

Writes a complete description of the ACMS_DATA facility selection to the file ACMS_DATA_FULL.LIST.

SHOW VERSION

Displays the version number of DECtrace running on your system.

Format

SHOW VERSION

Example

```
DECtrace> SHOW VERSION
DECtrace Version V1.0-0
DECtrace>
```

Displays the version number of DECtrace installed on the system—in this case, version V1.0.

SPAWN

Creates a subprocess of the current process. Allows you to temporarily exit from the DECtrace command environment without detaching from the DECtrace administration database.

Format

SPAWN [command]

Command Qualifiers	**Defaults**
/[NO]CARRIAGE_CONTROL	See text
/[NO]CLI	See text
/INPUT=file-spec	See text
/[NO]KEYPAD	/KEYPAD
/[NO]LOGICAL_NAMES	/LOGICAL_NAMES
/[NO]NOTIFY	/NONOTIFY
/OUTPUT=file-spec	/OUTPUT=SYS$OUTPUT
/PROCESS=subprocess-name	See text
/[NO]PROMPT[=string]	See text
/[NO]SYMBOLS	/SYMBOLS
/[NO]WAIT	/WAIT

Parameter

command
Specifies an optional command to be executed by the subprocess you are creating. If you specify the command parameter, you create a subprocess which executes the command and returns control to the DECtrace session when the command terminates. If you include the /INPUT qualifier with the command parameter, the subprocess reads the commands from the specified input file after the command executes. The command string cannot exceed 132 characters.

If you omit the command parameter, the SPAWN command creates a subprocess and attaches your terminal to it. You can return to your DECtrace session by logging out of the subprocess or by issuing the ATTACH/PARENT command. If you have created several subprocesses, you can switch between them using the ATTACH/IDENTIFICATION command.

Qualifiers

/CARRIAGE_CONTROL
/NOCARRIAGE_CONTROL
Determines whether carriage control or line feed characters (or both) are prefixed to the DCL-prompt string of the subprocess. The default is the current setting of the parent process.

/CLI[=cli]
/NOCLI
Specifies an alternate command language interpreter (CLI) for the subprocess to use. The default is the CLI the parent process uses.

The CLI you specify must be located in SYS$SYSTEM and have the file type EXE.

/INPUT=file-spec
Specifies an input file containing one or more commands for the spawned subprocess to execute. If you specify a command with an input file, the command is processed before the commands in the input file. The subprocess terminates when processing is complete. You cannot use wildcards in the file specification.

/KEYPAD (default)
/NOKEYPAD
Determines whether DCL keypad symbols and the current DCL keypad state are copied from the parent process to the subprocess. The default is to copy any key definitions or states (or both) you have established with the DEFINE /KEY command. Use the /NOKEYPAD qualifier if you do not want the key settings to be copied.

/LOGICAL_NAMES (default)
/NOLOGICAL_NAMES
Determines whether the system passes process logical names and logical name tables to the subprocess. The default is to copy all process logical names and logical name tables except those marked CONFINE or those created in executive or kernel mode.

/NOTIFY
/NONOTIFY (default)
Determines whether a message is sent to your terminal to notify you that your subprocess has been completed or aborted. Do not specify /NOTIFY unless you also specify the /NOWAIT qualifier.

/OUTPUT=file-spec
Specifies the output file to which the output of the SPAWN operation is to be written. When you specify /NOWAIT, you should use /OUTPUT to specify an output other than SYS$OUTPUT to prevent your terminal from being used by both processes simultaneously. By default, DECtrace directs the output to the current SYS$OUTPUT device (usually, your terminal).

/PROCESS=subprocess-name
Specifies the name of the subprocess to be created. The default name for the subprocess is USERNAME_#. The pound sign (#) denotes a unique number.

/PROMPT[=string]
Specifies the prompt string for the subprocess. If you specify /PROMPT but do not specify a string, the default prompt is displayed. The default is to copy the current prompt string from the parent process.

/SYMBOLS (default)
/NOSYMBOLS
Determines whether the system passes DCL global and local symbols to the subprocess.

/WAIT (default)
/NOWAIT
Controls whether the system waits until the subprocess is completed before allowing more commands to be issued by the parent process. The /NOWAIT qualifier allows you to enter more commands while the specified subprocess is running. When you specify /NOWAIT, you should also specify /OUTPUT to direct output to a file (rather than to your screen). This prevents your terminal from being used by both processes simultaneously.

Description

The SPAWN command creates a subprocess of your current DECtrace session—your parent process. The context of your DECtrace process is copied to the subprocess.

You can use the SPAWN command to leave the DECtrace command environment temporarily, to perform some non-DECtrace activities, and then return to your original DECtrace session. The advantage to using the SPAWN command is that you do not have to detach and reattach to the DECtrace administration database each time you need to leave the DECtrace command environment.

Example

```
$ COLLECT
DECtrace> SPAWN
$ SHOW DEVICE DUA2
Device                 Device         Error    Volume        Free  Trans Mnt
 Name                  Status         Count     Label       Blocks Count Cnt
MYVAX$DUA2:            Mounted            0     USER2       242686     2  25

$ LOGOUT
  Process SMITH_1 logged out at 10-FEB-1990 16:07:58.50
%EPC-S-SPAWN, Spawn successfully completed
DECtrace>
```

Creates a subprocess of the current process so that you can execute some non-DECtrace activities. In this case, the user checks the free space on a disk, then returns to the parent process.

STOP SYSTEM

Stops the DECtrace Registrar process and interrupts any active data collection. This command is usually performed as part of your system shutdown procedure. Stopping DECtrace is a management function that requires VMS OPER privilege to perform.

Format

STOP SYSTEM

Command Qualifier	**Default**
/[NO]ABORT	/NOABORT

Qualifier

/ABORT
/NOABORT (default)
Specifies whether DECtrace can abort active data collection on the system. If data collection is occurring on the system and you use the STOP SYSTEM command without the /ABORT qualifier, DECtrace issues an error message indicating that data collection must be interrupted, but it not stop the system.

To stop the system while data collection is occurring, you must use the /ABORT qualifier. In this situation, DECtrace issues a warning message that data collection will be interrupted and stops the system.

If no collections are currently active, DECtrace ignores the /[NO]ABORT qualifier.

Description

This command is *not* intended for normal interactive use. However, it is supplied for unusual cases where you want to stop the DECtrace system interactively. It is recommended that you stop the DECtrace system as part of your normal system shutdown procedure. You should add the following lines to your file SYS$MANAGER:SHUTDWN.COM:

```
$!Shutdown DECtrace and abort all active data collection
$ COLLECT STOP SYSTEM /ABORT
```

Note *You must stop the DECtrace* ***before*** *attempting to stop the Rdb/VMS monitor process.*

To restart DECtrace, issue the command:

```
$  @SYS$STARTUP:EPC$STARTUP.COM
```

Examples

1

```
$ COLLECT STOP SYSTEM
%EPC-I-STOPPED, System stopped at user's request.
$
```

Stops DECtrace if data collection is not currently active on the system.

2

```
$ COLLECT
DECtrace> STOP SYSTEM
%EPC-E-NOTSTOPPED, System was not stopped because data collection
is active.
%EPC-I-USEABORT, You must use the /ABORT qualifier to interrupt active
data collection.
```

Attempts to stop DECtrace while data collection is occurring. DECtrace issues an error message indicating that data collection is active on the system and that the system cannot be stopped without the /ABORT qualifier.

3

```
DECtrace> STOP SYSTEM /ABORT
%EPC-I-STOPPED, System stopped at user's request.
DECtrace> CTRL/Z
$
```

Interrupts all active data collection and shuts down the DECtrace registrar process.

8 DECtrace Service Routines

DECtrace provides a set of service routines to use when instrumenting an application program so that event data can be collected from it. These routines are to be embedded in the source code of the program. Chapter 5 describes how to correctly instrument your code with DECtrace service routine calls.

Table 8–1 lists the DECtrace service routine calls and summarizes their descriptions.

Table 8–1 DECtrace Service Routines

Routine Name	Description
EPC$DELETE_CONTEXT	Deletes the context associated with a particular thread (for multi-threaded facilities only).
EPC$END_EVENT	Records the end of a duration event to the data collection file.
EPC$END_EVENTW	Records the end of a duration event to the data collection file and waits for processing to complete before returning.
EPC$EVENT	Records the occurrence of a point event to the data collection file.
EPC$EVENTW	Records the occurrence of a point event to the data collection file and waits for processing to complete before returning.
EPC$INIT	Registers a facility with DECtrace to enable data collection on this facility.

(continued on next page)

Table 8–1 (Cont.) DECtrace Service Routines

Routine Name	Description
EPC$SET_CONTEXT	Sets the context for a new or existing thread so resource utilization items can be collected (for multi-threaded facilities only).
EPC$START_EVENT	Records the start of a duration event to the data collection file.
EPC$START_EVENTW	Records the start of a duration event to the data collection file and waits for processing to complete before returning.

The DECtrace service routines are similar to the VMS system services. Any routines that perform I/O have two types: synchronous (W) and asynchronous. A synchronous routine ends in W (for example, EPC$START_EVENTW) whereas the asynchronous version has no W (for example, EPC$START_EVENT). The routines that record data collection records in the data collection file are EPC$EVENT(W), EPC$START_EVENT(W), and EPC$END_EVENT(W). All DECtrace routines operate in executive mode and are AST reentrant.

8.1 Error Handling

Error reporting is taken care of as follows. Each routine returns a longword condition value informing the caller of the completion status of the call. An optional quadword argument containing the completion status for the asynchronous part of the calls can be returned to the caller. In addition, all errors encountered during data collection are logged to the system-wide or cluster-wide history database.

Two types of errors are returned: DECtrace errors and system service errors. DECtrace errors are prefaced with EPC$. If an unexpected error occurs during a system routine, DECtrace returns the actual system code. These system errors usually result from low process quotas or system parameters. EPC$INIT calls the following system routines: GETSYI, GETTIM, ENQ, DEQ, SETEF, CLREF, CREMBX, GETJPI, GETDVI, QIO, EXPREG, and DCLAST. The DECtrace start, end, and point event routines call the following system routines: GETTIM, ENQ, DEQ, SETEF, CLREF, QIO, EXPREG, and DCLAST.

Each DECtrace service routine has a set of return values associated with it. See the description of each service routine for a complete list of return values. Most of these values are self explanatory, but the following values are not as obvious as the rest:

- The EPC$_DISABLED return value has several different meanings depending on the context it is returned in. First, this value is returned if the facility and version do not match any active collection. Second, if this facility had been collecting data within the context of the current image and an error occurred on an earlier DECtrace call, EPC$_DISABLED will be returned on subsequent calls to DECtrace. Finally, this value will be returned if data collection has been disabled by defining the EPC$DISABLED logical. See Section 9.5 for more information on disabling data collection.
- The EPC$_EVNTNOTCOL return value indicates that the facility and version match the criteria of an active collection, but the current event was not specified in the selection criteria.
- The EPC$_FACNOTCOL return value indicates that the process has registered another facility which is actively collecting data, but the current facility does not match the selection criteria.
- The EPC$_INIT2 return value indicates that the current facility and version have already registered once within this image activation. This represents an instrumentation error in the application. EPC$INIT should only be called once by each facility within an image activation.
- The EPC$_NOREGEXISTS return value indicates that the DECtrace Registrar process (EPC$REGISTRAR) is not running on the local node. Use the SYS$STARTUP:EPC$STARTUP.COM procedure to restart the Registrar process. See Section 9.2 for more information on starting and stopping the Registrar process.
- The EPC$_NOTINSTALLED return value indicates that the DECtrace software is not installed on the system.

EPC$DELETE_CONTEXT

EPC$DELETE_CONTEXT deletes all associated context for the given context variable.

Format

EPC$DELETE_CONTEXT [efn] ,context-variable [,status] [,astadr ,astparam]

Returns

VMS Usage:	**cond_value**
type:	**longword (unsigned)**
access:	**write only**
mechanism:	**by value**

Arguments

efn

VMS Usage:	**ef_number**
type:	**longword (unsigned)**
access:	**read only**
mechanism:	**by value**

The VMS event flag to be set when EPC$DELETE_CONTEXT completes. The **efn** argument is a longword containing the address of this flag.

Upon initiation, EPC$DELETE_CONTEXT clears the specified event flag (or event flag 0 if **efn** was not specified). When EPC$DELETE_CONTEXT returns, it sets the specified event flag (or event flag 0).

context-variable

VMS Usage:	**context**
type:	**longword (unsigned)**
access:	**modify**
mechanism:	**by reference**

The address of the longword that contains the context variable to delete. EPC$DELETE_CONTEXT sets the longword value to zero.

status

VMS Usage:	**io_status_block**
type:	**quadword (unsigned)**
access:	**write only**
mechanism:	**by reference**

When you specify the **status** argument, EPC$DELETE_CONTEXT sets the first longword to zero at initiation. The second longword is always zero. Upon completion, a condition value returns in the first longword.

astadr

VMS Usage:	**ast_procedure**
type:	**procedure entry mask**
access:	**call without stack unwinding**
mechanism:	**by reference**

The address of the entry mask of the AST service routine to be executed when EPC$DELETE_CONTEXT completes.

If you specify the **astadr** argument, the AST routine executes in the same access mode as the caller of EPC$DELETE_CONTEXT.

astparam

VMS Usage:	**user_arg**
type:	**longword (unsigned)**
access:	**read only**
mechanism:	**by value**

The AST parameter to be passed to the AST service routine specified by the **astadr** argument. The **astparam** argument is a longword parameter.

Include this argument only when you specify the **astadr** argument.

Description

The EPC$DELETE_CONTEXT routine deletes the context variable. All associated context for DECtrace to collect resource utilization items for a particular context variable is deleted.

The value of the **context-variable** argument is set to zero upon completion.

A facility calls the EPC$DELETE_CONTEXT service only if an associated EPC$SET_CONTEXT call was previously issued and when execution of a thread completes to its entirety.

Return Values

EPC$_BADASTADR	Bad AST address specified.
EPC$_BADASTPRM	Bad AST parameter specified.
EPC$_BADSTATUS	Bad status argument passed.
EPC$_BADTHREAD	Bad thread argument passed.
EPC$_DISABLED	Collection has been disabled.
EPC$_INSARGS	Insufficient arguments specified.
EPC$_NOTHREAD	No thread argument was passed.
EPC$_NOTINSTALL	DECtrace software is not installed.
EPC$_SUCCESS	This call was successful.

EPC$END_EVENT

EPC$END_EVENT records the end of an event to the data collection file.

The EPC$END_EVENT routine completes asynchronously. It returns control to the caller after initiating the $END_EVENT processing, without waiting for I/O processing to complete.

Format

EPC$END_EVENT [efn] ,facility ,event-id ,handle [,context-variable] [,event-rec] [,status] [,astadr ,astparam]

Returns

VMS Usage:	**cond_value**
type:	**longword (unsigned)**
access:	**write only**
mechanism:	**by value**

Arguments

efn

VMS Usage:	**ef_number**
type:	**longword (unsigned)**
access:	**read only**
mechanism:	**by value**

The VMS event flag to be set when EPC$END_EVENT completes. The **efn** argument is a longword containing the address of this flag.

Upon initiation, EPC$END_EVENT clears the specified event flag (or event flag 0 if **efn** was not specified). When EPC$END_EVENT returns, it sets the specified event flag (or event flag 0).

facility

VMS Usage:	**longword_unsigned**
type:	**longword (unsigned)**
access:	**read only**
mechanism:	**by value**

The facility number for the calling facility. Facilities 1 through 2047 are reserved for Digital. Facilities 2048 through 4097 are user-defined.

event-id

VMS Usage:	**longword_unsigned**
type:	**longword (unsigned)**
access:	**read only**
mechanism:	**by value**

The event identifier for the event.

handle

VMS Usage:	**context**
type:	**longword (unsigned)**
access:	**write only**
mechanism:	**by reference**

The address of the longword to retrieve a process unique event handle for this particular event.

The supplied longword **handle** is the same **handle** that was returned from the corresponding EPC$START_EVENT call for an event instance.

context-variable

VMS Usage:	**context**
type:	**longword (unsigned)**
access:	**read only**
mechanism:	**by reference**

The address of the longword used to identify the context in which this call appears. When specified, the value of the longword corresponds to the **context-variable** argument returned from the call to EPC$SET_CONTEXT.

event-rec

VMS Usage:	**vector_byte_signed**
type:	**any binary or ASCII data**
access:	**read only**
mechanism:	**by descriptor—fixed length descriptor**

The address of a descriptor pointing to a record buffer containing the item values for an event. The item values are facility specific; they do not include any resource utilization items. DECtrace handles the resource item values transparently. As a result, you do not need to specify the **event-rec** on the EPC$END_EVENT call if you want only the set of standard resource utilization items.

status

VMS Usage:	**io_status_block**
type:	**quadword (unsigned)**
access:	**write only**
mechanism:	**by reference**

When you specify the status argument, EPC$END_EVENT sets the first longword to zero at initiation. The second longword is always zero. Upon completion, a condition value returns in the first longword.

astadr

VMS Usage:	**ast_procedure**
type:	**procedure entry mask**
access:	**call without stack unwinding**
mechanism:	**by reference**

The address of the entry mask of the AST service routine to be executed when EPC$END_EVENT completes.

If **astadr** is specified, the AST routine executes in the same access mode as the caller of EPC$END_EVENT.

astparam

VMS Usage:	**user_arg**
type:	**longword (unsigned)**
access:	**read only**
mechanism:	**by value**

AST parameter to be passed to the AST service routine specified by the **astadr** argument. The **astparam** argument is a longword parameter.

Include this argument only when you specify the **astadr** argument.

Description

The EPC$END_EVENT routine enables a given facility to log an end event record to the data collection file. This service is called at the end of a duration event with various items determined by the item flags list for a given facility.

For performance reasons, DECtrace does not write every record directly to the data collection file. Instead, records are buffered and flushed to disk when the buffer is full. The EPC$END_EVENT routine completes asynchronously. It returns to the caller after copying the end event record to the buffer without waiting for any buffer flushes to complete.

Return Values

EPC$_BADASTADR	Bad AST address specified.
EPC$_BADASTPRM	Bad AST parameter specified.
EPC$_BADEVNT	Bad event ID argument passed.
EPC$_BADEVNTREC	Bad event record argument passed.
EPC$_BADFAC	Bad facility code passed.
EPC$_BADHANDL	Bad handle argument passed.
EPC$_BADSTATUS	Bad status argument passed.
EPC$_BADTHREAD	Bad thread argument passed.
EPC$_DCFCORRUPT	Data capture file is corrupt.
EPC$_DISABLED	Collection has been disabled.
EPC$_EVTNOTCOL	Event is not being collected.
EPC$_FACNOTCOL	Facility is not being collected.
EPC$_ILLBUFLEN	The record buffer was larger than the maximum.
EPC$_INSARGS	Insufficient arguments specified.
EPC$_NODCFEXISTS	The specified data capture file does not exist.
EPC$_NOEVNT	No event ID argument specified.
EPC$_NOFAC	No facility argument specified.
EPC$_NOHANDL	No handle argument passed.
EPC$_NOTINSTALL	DECtrace software is not installed.
EPC$_SUCCESS	This call was successful.

EPC$END_EVENTW

EPC$END_EVENTW records the end of an event to the data collection file. This service is called at the end of a duration event with various items determined by the item flags list for a given facility.

The EPC$END_EVENTW routine completes synchronously. It returns control to the caller when the $END_EVENTW processing completes.

Format

EPC$END_EVENTW [efn] ,facility ,event-id ,handle [,context-variable] [,event-rec] [,status] [,astadr ,astparam]

Description

For asynchronous completion, use the EPC$END_EVENT routine. EPC$END_EVENT returns to the caller after initiating the processing without waiting for I/O to complete.

For performance reasons, DECtrace does not write every record directly to the data collection file. Instead, records are buffered and flushed to disk when the buffer is full. The EPC$END_EVENTW routine completes synchronously. That is, if a buffer flush needs to be performed, it waits for I/O to complete before returning to the caller.

In all other respects, EPC$END_EVENTW is identical to EPC$END_EVENT. Refer to EPC$END_EVENT for all other information about the EPC$END_EVENTW service routine.

EPC$EVENT

EPC$EVENT records the occurrence of a point event to the data collection file.

The EPC$EVENT routine completes asynchronously. It returns control to the caller after initiating the $EVENT processing without waiting for I/O processing to complete.

Format

EPC$EVENT [efn] ,facility ,event-id [,context-variable] [,event-rec] [,status] [,astadr ,astparam]

Returns

VMS Usage:	**cond_value**
type:	**longword (unsigned)**
access:	**write only**
mechanism:	**by value**

Arguments

efn

VMS Usage:	**ef_number**
type:	**longword (unsigned)**
access:	**read only**
mechanism:	**by value**

The VMS event flag to be set when EPC$EVENT completes. The **efn** argument is a longword containing the address of this flag.

Upon initiation, EPC$EVENT clears the specified event flag (or event flag 0 if **efn** was not specified). When EPC$EVENT returns, it sets the specified event flag (or event flag 0).

facility

VMS Usage:	**longword_unsigned**
type:	**longword (unsigned)**
access:	**read only**
mechanism:	**by value**

The facility number for the calling facility. Facilities 1 through 2047 are reserved for Digital. Facilities 2048 through 4097 are user-defined.

event-id

VMS Usage:	**longword_unsigned**
type:	**longword (unsigned)**
access:	**read only**
mechanism:	**by value**

The event identifier for an event.

context-variable

VMS Usage:	**context**
type:	**longword (unsigned)**
access:	**read only**
mechanism:	**by reference**

The address of the longword used to identify the context variable. When specified, the value of the longword corresponds to the **context-variable** argument returned from the call to EPC$SET_CONTEXT.

event-rec

VMS Usage:	**vector_byte_signed**
type:	**any binary or ASCII data**
access:	**read only**
mechanism:	**by descriptor—fixed length descriptor**

The address of a descriptor pointing to a record buffer containing the item values for an event. The item values are facility specific; they do not include the resource utilization items. DECtrace handles the resource item values transparently. As a result you do not need to specify the **event-rec** on the EPC$EVENT call if you only want the set of standard resource utilization items.

status

VMS Usage:	**io_status_block**
type:	**quadword (unsigned)**
access:	**write only**
mechanism:	**by reference**

When you specify the **status** argument, EPC$EVENT sets the first longword to zero at initiation. The second longword is always zero. Upon completion, a condition value returns in the first longword.

astadr

VMS Usage:	**ast_procedure**
type:	**procedure entry mask**
access:	**call without stack unwinding**

mechanism: **by reference**

The address of the entry mask of the AST service routine to be executed when EPC$EVENT completes.

If the **astadr** is specified, the AST routine executes in the same access mode as the caller of EPC$EVENT.

astparam
VMS Usage: **user_arg**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

The AST parameter to be passed to the AST service routine specified by the **astadr** argument. The **astparam** argument is a longword parameter.

Include this argument only when you specify the **astadr** argument.

Description

EPC$EVENT routine enables a given facility to record the occurrence of a point event to the data collection file.

For performance reasons, DECtrace does not write every record directly to the data collection file. Instead, records are buffered and flushed to disk when the buffer is full. The EPC$EVENT routine completes asynchronously. That is, it returns to the caller after copying the point event record to the buffer without waiting for any buffer flushes to complete.

Return Values

EPC$_BADASTADR	Bad AST address specified.
EPC$_BADASTPRM	Bad AST parameter specified.
EPC$_BADEVNT	Bad event ID argument passed.
EPC$_BADEVNTREC	Bad event record argument passed.
EPC$_BADFAC	Bad facility code passed.
EPC$_BADSTATUS	Bad status argument passed.
EPC$_BADTHREAD	Bad thread argument passed.
EPC$_DCFCORRUPT	Data capture file is corrupt.

EPC$_DISABLED	Collection has been disabled.
EPC$_EVTNOTCOL	Event is not being collected.
EPC$_FACNOTCOL	Facility is not being collected.
EPC$_ILLBUFLEN	The record buffer was larger than the maximum.
EPC$_INSARGS	Insufficient arguments specified.
EPC$_NODCFEXISTS	The specified data capture file does not exist.
EPC$_NOEVNT	No event ID argument specified.
EPC$_NOFAC	No facility argument specified.
EPC$_NOTINSTALL	DECtrace software is not installed.
EPC$_SUCCESS	This call was successful.

EPC$EVENTW

EPC$EVENTW records the occurrence of a point event to the data collection file and waits for I/O processing to complete before returning control.

The EPC$EVENTW routine completes synchronously. It returns to the caller when the $EVENTW processing completes.

Format

EPC$EVENTW [efn] ,facility ,event-id [,context-variable] [,event-rec] [,status] [,astadr ,astparam]

Description

For asynchronous completion, use the EPC$EVENT service; EPC$EVENT returns to the caller after initiating the processing without waiting for I/O to complete.

For performance reasons, DECtrace does not write every record directly to the data collection file. Instead, records are buffered and flushed to disk when the buffer is full. The EPC$EVENTW routine completes synchronously. That is, if a buffer flush needs to be performed, it will wait for the I/O to complete before returning to the caller.

In all other respects, EPC$EVENTW is identical to EPC$EVENT. Refer to EPC$EVENT for all other information about the EPC$EVENTW service routine.

EPC$INIT

EPC$INIT registers a facility with DECtrace.

Format

EPC$INIT [efn] ,facility ,version [,registration_id] [,eventflgs] [,itemflgs] [,status] [,astadr ,astparam]

Returns

VMS Usage:	**cond_value**
type:	**longword (unsigned)**
access:	**write only**
mechanism:	**by value**

Arguments

efn

VMS Usage:	**ef_number**
type:	**longword (unsigned)**
access:	**read only**
mechanism:	**by value**

The VMS event flag to be set when EPC$INIT completes. The **efn** argument is a longword containing the address of this flag.

Upon initiation, EPC$INIT clears the specified event flag (or event flag 0 if **efn** was not specified). When EPC$INIT returns, it sets the specified event flag (or event flag 0).

facility

VMS Usage:	**longword_unsigned**
type:	**longword (unsigned)**
access:	**read only**
mechanism:	**by value**

The facility number for the calling facility. Facilities 1 through 2047 are reserved for Digital. Facilities 2048 through 4097 are user-defined.

version

VMS Usage:	**char_string**
type:	**character-coded text string**
access:	**read only**
mechanism:	**by descriptor**

The string, up to 10 characters, identifying the current version of the facility. Note that the string must match identically with that specified in the facility definition.

registration_id

VMS Usage:	**char_string**
type:	**character-coded text string**
access:	**read only**
mechanism:	**by descriptor**

A string of up to 255 characters containing the identifier this facility wants to associate with this VMS process. This identifier is "registered" in the DECtrace administration database along with other information for this process. The registration identifier gives the user the ability to schedule data collection for a group of processes. For example, if ACMS uses this identifier to describe a particular application, the user can then choose to schedule data collection for all processes that belong to that particular application, rather than collect data on all ACMS applications.

The **registration-id** can contain the following characters:

- A through Z
- a through z
- 0 through 9
- Underscore (_)
- Hyphen (-)

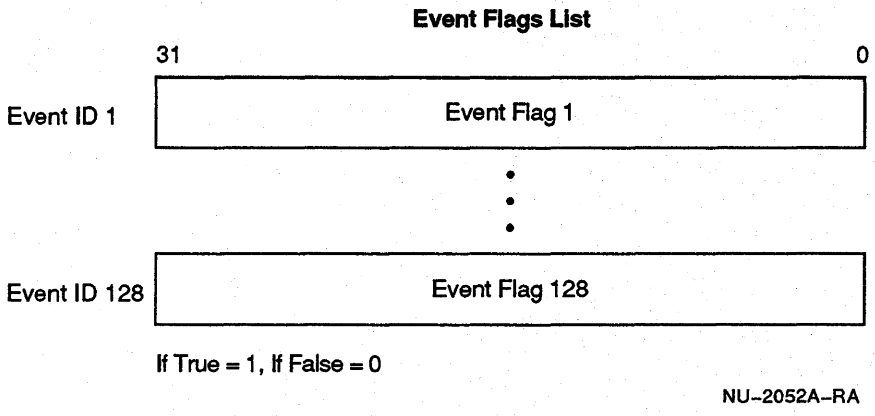
eventflgs

VMS Usage:	**vector_longword_unsigned**
type:	**longword (unsigned)**
access:	**write only**
mechanism:	**by reference**

A list of 128 longword Boolean values. Each list element corresponds to an event identifier and designates whether data collection is enabled for a particular event.

A facility needs only one copy of the event flags.

The following figure shows the physical structure of the event flags:

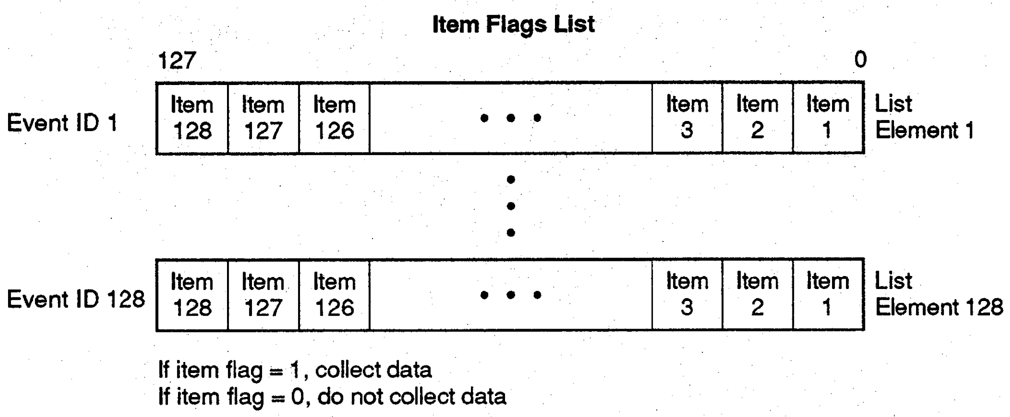


Itemflgs
VMS Usage: **item_flags_list**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**

A list of 128 item flags. Each set of item flags, referred to as a list element, is 128 bits and corresponds to a particular event. The flags designate which items are to be collected for a particular event. If bit 0 is set for item flags 1, then item 1 is to be collected for event 1.

The facility needs only one copy of the item flags.

The following figure shows the physical structure of the item flags list:



NU-2053A-RA

status

VMS Usage:	**io_status_block**
type:	**quadword (unsigned)**
access:	**write only**
mechanism:	**by reference**

When you specify the **status** argument, EPC$INIT sets the first longword to zero at initiation. The second longword is always zero. Upon completion, a condition value returns in the first longword.

astadr

VMS Usage:	**ast_procedure**
type:	**procedure entry mask**
access:	**call without stack unwinding**
mechanism:	**by reference**

The address of the entry mask of the AST service routine to be executed when EPC$INIT completes.

If the **astadr** is specified, the AST routine executes in the same access mode as the caller of EPC$INIT.

astparam

VMS Usage:	**user_arg**
type:	**longword (unsigned)**
access:	**read only**
mechanism:	**by value**

The AST parameter to be passed to the AST service routine specified by the **astadr** argument. The **astparam** argument is a longword parameter.

Include this argument only when you specify the **astadr** argument.

Description

The EPC$INIT routine registers a facility with DECtrace. By registering, active and pending collections are enabled for this facility (provided that data collection is scheduled using the SCHEDULE command).

The EPC$INIT routine completes in both a synchronous and asynchronous nature. The initial values of the **eventflgs** and **itemflgs** arguments, if specified, are returned synchronously, which allows the facility to determine which events and associated items are to be collected immediately following the call to EPC$INIT. However, the values of the **eventflgs** and **itemflgs** can change as collections become active. The values of the **eventflgs** and **itemflgs** change to reflect the values in the corresponding facility selection for the data collection.

As collections become active, the DECtrace registrar process notifies the application process of the event and item flags for that specific collection. Because of the communication involved and the asynchronous nature by which you can schedule data collection, the values of the event and item flags can change asynchronously.

The **status** argument is returned when EPC$INIT completes.

Return Values

EPC$_BADEVNTFLGS	Bad facility event flags passed.
EPC$_BADFAC	Bad facility code passed.
EPC$_BADFACVER	Bad facility version passed.
EPC$_BADITMFLGS	Bad facility item flags passed.
EPC$_BADREGID	Bad facility registration group ID passed.
EPC$_DISABLED	Data collection disabled.
EPC$_FACVERREQ	Facility version required.
EPC$_INIT2	EPC$INIT called twice.
EPC$_NODCFEXISTS	No data capture file exists.

EPC$_NOREGEXISTS	No registrar process exists on this node.
EPC$_NOTINSTALL	DECtrace software is not installed.
EPC$_SUCCESS	Routine completed successfully.

EPC$SET_CONTEXT

EPC$SET_CONTEXT records a context variable for a new or existing thread so that resource utilization items can be collected for multi-threaded facilities.

Format

EPC$SET_CONTEXT [efn] ,context-variable [,status] [,astadr ,astparam]

Returns

VMS Usage:	**cond_value**
type:	**longword (unsigned)**
access:	**write only**
mechanism:	**by value**

Arguments

efn

VMS Usage:	**ef_number**
type:	**longword (unsigned)**
access:	**read only**
mechanism:	**by value**

The VMS event flag to be set when EPC$SET_CONTEXT completes. The **efn** argument is a longword containing the address of this flag.

Upon initiation, EPC$SET_CONTEXT clears the specified event flag (or event flag 0 if **efn** was not specified). When EPC$SET_CONTEXT returns, it sets the specified event flag (or event flag 0).

context-variable

VMS Usage:	**context**
type:	**longword(unsigned)**
access:	**modify**
mechanism:	**by reference**

The address of the longword that contains the context variable to which context should be set.

If a zero value is passed at the address specified, then EPC$SET_CONTEXT returns a value that identifies the new thread context. If the address contains a non-zero context variable, then a context change record is written to the data collection file.

status

VMS Usage:	**io_status_block**
type:	**quadword (unsigned)**
access:	**write only**
mechanism:	**by reference**

When you specify the **status** argument, EPC$SET_CONTEXT sets the first longword to zero at initiation. The second longword is always zero. Upon completion, a condition value returns in the first longword.

astadr

VMS Usage:	**ast_procedure**
type:	**procedure entry mask**
access:	**call without stack unwinding**
mechanism:	**by reference**

The address of the entry mask of the AST service routine to be executed when EPC$SET_CONTEXT completes. The **astadr** is the address of the entry mask of this routine.

If you specify the **astadr**, the AST routine executes in the same access mode as the caller of EPC$SET_CONTEXT.

astparam

VMS Usage:	**user_arg**
type:	**longword (unsigned)**
access:	**read only**
mechanism:	**by value**

The AST parameter to be passed to the AST service routine specified by the **astadr** argument. The **astparam** argument is a longword parameter.

Include this argument only when you specify the **astadr** argument.

Description

The EPC$SET_CONTEXT routine sets context for a new thread or an existing thread so that resource utilization items can be collected for multi-threaded facilities. This routine is called by a facility when a thread context is initially created or reentered (by way of a context switch) when resource utilization items are to be collected on a per-thread basis.

If a zero value is passed at the address specified, then EPC$SET_CONTEXT returns a value that identifies the new thread context.

The associated EPC$DELETE_CONTEXT routine must be called by a multi-threaded facility when a thread executes to its entirety in order to delete context information that DECtrace maintains on behalf of the facility.

Return Values

EPC$_BADASTADR	Bad AST address specified.
EPC$_BADASTPRM	Bad AST parameter specified.
EPC$_BADSTATUS	Bad status argument passed.
EPC$_BADTHREAD	Bad thread argument passed.
EPC$_DISABLED	Collection has been disabled.
EPC$_INSARGS	Insufficient arguments specified.
EPC$_NOTHREAD	No thread argument was passed.
EPC$_NOTINSTALL	DECtrace software is not installed.
EPC$_SUCCESS	This call was successful.

EPC$START_EVENT

EPC$START_EVENT records the start of an event to the data collection file.

The EPC$START_EVENT routine completes asynchronously. It returns to the caller after initiating the $START_EVENT processing without waiting for I/O processing to complete.

Format

EPC$START_EVENT [efn] ,facility ,event-id ,handle [,context-variable] [,event-rec] [,status] [,astadr ,astparam]

Returns

VMS Usage:	**cond_value**
type:	**longword (unsigned)**
access:	**write only**
mechanism:	**by value**

Arguments

efn

VMS Usage:	**ef_number**
type:	**longword (unsigned)**
access:	**read only**
mechanism:	**by value**

The VMS event flag to be set when EPC$START_EVENT completes. The **efn** argument is a longword containing the address of this flag.

Upon initiation, EPC$START_EVENT clears the specified event flag (or event flag 0 if **efn** was not specified). When EPC$START_EVENT returns, it sets the specified event flag (or event flag 0).

facility

VMS Usage:	**longword_unsigned**
type:	**longword (unsigned)**
access:	**read only**
mechanism:	**by value**

The facility number for the calling facility. Facilities 1 through 2047 are reserved for Digital. Facilities 2048 through 4097 are user-defined.

event-id

VMS Usage:	**longword_unsigned**
type:	**longword (unsigned)**
access:	**read only**
mechanism:	**by value**

The event identifier for an event.

handle

VMS Usage:	**context**
type:	**longword (unsigned)**
access:	**write only**
mechanism:	**by reference**

The address of the longword to store a process unique event handle for this particular event. The **handle** is used in the corresponding EPC$END_EVENT call for an event.

context-variable

VMS Usage:	**context**
type:	**longword (unsigned)**
access:	**read only**
mechanism:	**by reference**

The address of the longword used to identify the context in which this call appears. When specified, the value of the longword corresponds to the **context-variable** argument returned from the call to EPC$SET_CONTEXT.

event-rec

VMS Usage:	**vector_byte_signed**
type:	**any binary or ASCII data**
access:	**read only**
mechanism:	**by descriptor**

The address of a descriptor pointing to a record buffer containing the item values for an event. The item values are facility specific; they do not include the resource utilization items. DECtrace handles the resource item values transparently. As a result, you do not need to specify the **event-rec** on the EPC$START_EVENT call if you want only the set of standard resource utilization items.

status

VMS Usage:	**io_status_block**
type:	**quadword (unsigned)**
access:	**write only**

mechanism: **by reference**

When you specify the **status** argument, EPC$START_EVENT sets the first longword to zero at initiation. The second longword is always zero. Upon completion, a condition value returns in the first longword.

astadr

VMS Usage:	**ast_procedure**
type:	**procedure entry mask**
access:	**call without stack unwinding**
mechanism:	**by reference**

The address of the entry mask of the AST service routine to be executed when EPC$START_EVENT completes.

If you specify the **astadr**, the AST routine executes in the same access mode as the caller of EPC$START_EVENT.

astparam

VMS Usage:	**user_arg**
type:	**longword (unsigned)**
access:	**read only**
mechanism:	**by value**

The AST parameter to be passed to the AST service routine specified by the **astadr** argument. The **astparam** argument is a longword parameter.

Include this argument only when you specify the **astadr** argument.

Description

The EPC$START_EVENT routine enables a given facility to log the start of an event to the data collection file. This routine is called at the beginning of a duration event that can have various facility-specific items associated with it. The items recorded are determined by the item flags list for the facility. A call to EPC$END_EVENT follows this call where the event ends.

For performance reasons, DECtrace does not write every record directly to the data collection file. Instead, records are buffered and flushed to disk when the buffer is full. The EPC$START_EVENT routine completes asynchronously. That is, it returns to the caller after copying the start event record to the buffer without waiting for any buffer flushes to complete.

Return Values

EPC$_BADASTADR	Bad AST address specified.
EPC$_BADASTPRM	Bad AST parameter specified.
EPC$_BADEVNT	Bad event ID argument passed.
EPC$_BADEVNTREC	Bad event record argument passed.
EPC$_BADFAC	Bad facility code passed.
EPC$_BADHANDL	Bad handle argument passed.
EPC$_BADSTATUS	Bad status argument passed.
EPC$_BADTHREAD	Bad thread argument passed.
EPC$_DCFCORRUPT	Data capture file is corrupt.
EPC$_DISABLED	Collection has been disabled.
EPC$_EVTNOTCOL	Event is not being collected.
EPC$_FACNOTCOL	Facility is not being collected.
EPC$_ILLBUFLEN	The record buffer was larger than the maximum.
EPC$_INSARGS	Insufficient arguments specified.
EPC$_NODCFEXISTS	The specified data capture file does not exist.
EPC$_NOEVNT	No event ID argument specified.
EPC$_NOFAC	No facility argument specified.
EPC$_NOHANDL	No handle argument passed.
EPC$_NOTINSTALL	DECtrace software is not installed.
EPC$_SUCCESS	This call was successful.

EPC$START_EVENTW

EPC$START_EVENTW records the start of a duration event to the data collection file.

The EPC$START_EVENTW routine completes synchronously. It returns to the caller after all $START_EVENTW processing completes.

Format

EPC$START_EVENTW [efn] ,facility ,event-id ,handle [,thread] [,event-rec] [,status] [,astadr ,astparam]

Description

For asynchronous completion, use the EPC$START_EVENT service; EPC$START_EVENT returns control to the caller after initiating the processing, without waiting for I/O processing to complete.

For performance reasons, DECtrace does not write every record directly to the data collection file. Instead, records are buffered and flushed to disk when the buffer is full. The EPC$START_EVENTW routine completes synchronously; if a buffer flush needs to be performed, it waits for the I/O to complete before returning to the caller.

In all other respects, EPC$START_EVENTW is identical to EPC$START_EVENT. Refer to EPC$EVENT for all other information about the EPC$START_EVENTW routine.

9 System Management Tasks

This chapter describes the system management functions associated with using DECtrace.

9.1 Required Account and Process Quotas

You must make sure that the appropriate user accounts have sufficient quotas to be able to use DECtrace. If you typically format very large collection files (over 50,000 blocks) and generate reports based on large formatted databases, you can improve performance by increasing several of your account quotas. Table 9–1 summarizes the required and optional user account quotas.

Table 9–1 User Account Quotas for Using DECtrace

Account Quota	Normal Use	Formatting and Reporting on Large Files
ASTLM	24	
BIOLM	20	
BYTLM	20,480	34,810
DIOLM	20	
ENQLM	1,800	10,000
FILLM	50	
PGFLQUO	20,000	75,000
PRCLM	1	
WSEXTENT	2048	
WSQUOTA	1024	

User account quotas are stored in the file SYSUAF.DAT. Use the VMS Authorize Utility to verify and change user account quotas. First set your directory to SYS$SYSTEM and then run AUTHORIZE:

```
$ SET DEFAULT SYS$SYSTEM
$ RUN AUTHORIZE
UAF>
```

At the UAF> prompt, use the SHOW command with an account name to check a particular account. For example:

```
UAF> SHOW SMITH
```

To change a quota, use the MODIFY command at the UAF> prompt. MODIFY has the following format:

MODIFY account-name */quota-name=NNN*

The following example changes the FILLM quota for the SMITH account and then exits from the utility:

```
UAF>MODIFY SMITH /FILLM=50
UAF>EXIT
```

After you exit from the utility, the system displays messages indicating whether or not changes were made. Once you have made the changes, users must log out and log in again for the new quotas to take effect.

For more information on modifying account quotas, see the *VMS Authorize Utility Manual.*

9.2 Controlling DECtrace

A part of DECtrace called the **Registrar process** is designed to be active on your system or cluster at all times. This process handles all communication between your application programs and DECtrace. Without this process, applications instrumented with DECtrace calls cannot register with DECtrace. In addition, you cannot schedule data collection without the Registrar process.

9.2.1 Starting DECtrace

Generally, you should start the DECtrace Registrar process as part of your normal system startup procedure. Note that you must start the Registrar process *after* activating the Rdb/VMS monitor process on your system.

Add the DECtrace startup procedure to the system startup command file on each node that you want DECtrace to run on:

```
$! Start the Rdb/VMS monitor process
$ @SYS$STARTUP:RMONSTART
$! Start DECtrace
$ @SYS$STARTUP:EPC$STARTUP
```

You can also restart the Registrar process interactively with the following command:

```
$  @SYS$STARTUP:EPC$STARTUP
```

9.2.2 Stopping DECtrace

Digital recommends that you stop DECtrace as part of your normal system shutdown procedure. Note that you must stop the DECtrace Registrar process *before* stopping the Rdb/VMS monitor process. If you attempt to stop the Rdb/VMS monitor first, both processes will hang and you will have to use the VMS STOP/ID command to cancel the EPC$REGISTRAR process.

Add the DECtrace shutdown command to SYS$MANAGER:SYSHUTDWN.COM on each node that DECtrace is running on:

```
$! Shutdown DECtrace and abort all active data collection
$ COLLECT STOP SYSTEM/ABORT
$! Shutdown the Rdb/VMS monitor process
$ @SYS$MANAGER:RMONSTOP
```

You can also stop the Registrar process interactively with the following command:

```
$  COLLECT STOP SYSTEM/ABORT
```

Note that when the Registrar process is stopped, any processes that are actively collecting data continue to record data to the data collection file(s). Data collection continues until either the image terminates or the Registrar is restarted and the collection ends normally.

9.2.3 Recovering From a System Crash

After a system crash, the DECtrace registrar process should restart automatically as part of your normal system startup procedure.

If a collection is active when the system goes down, data collection resumes so long as the registrar process is restarted before the scheduled end-time of the collection. If a collection is pending when the system goes down, data collection begins as scheduled so long as the registrar process restarts before the scheduled end-time. For example, consider the following collections scheduled on node MYVAX1:

```
24-JUL-1989 08:45          Scheduled Collections                    Page 1
Brief Report                                                   DECtrace V1.0

Collection Schedule for node MYVAX1

    Selection Name   Collection Name  Start            End
    ---------------  ---------------  ---------------  ---------------
->  MY_SELECTION     MORNING_DATA     24-JUL-89 08:00  24-JUL-89 10:00
    TOOLS_SELECTION  JOB123           24-JUL-89 10:00  24-JUL-89 12:00
    MY_SELECTION     AFTERNOON_DATA   24-JUL-89 12:00  24-JUL-89 16:00
```

If MYVAX1 crashes at 09:00 and comes back at 09:30, the collection MORNING_DATA resumes data collection for the remaining 15 minutes of the scheduled collection interval. The pending collections: JOB123 and AFTERNOON_DATA are unaffected by the system crash and start as scheduled.

If MYVAX1 crashes at 09:00 and comes back at 10:15, the collection JOB123 starts and runs until its normal completion time. The pending collection: AFTERNOON_DATA is unaffected by the system crash. Note that the collection MORNING_DATA successfully collected data from 08:00 until the crash occurred at 09:00. The partial data capture file(s) produced by the collection can be formatted and reported on as if the collection had run to completion.

9.3 Installing Facilities on Your System

When you perform a product installation on your system, there are several cases where this could affect your use of DECtrace.

9.3.1 Installing New Facilities

When you install a product that has just added DECtrace support or when you install a supporting product for the first time, there is nothing that you need to do for DECtrace. The product's installation procedure automatically inserts a new facility definition into the DECtrace administration database.

9.3.2 Installing New Versions of Existing Facilities

When you install a product whose previous version also supported DECtrace, you need to update any of your facility selections which specifically reference the previous version of the product. The product's installation procedure automatically inserts a new facility definition into the DECtrace administration database, but unless you change your facility selections to use the correct version number for the new product version, you will not be able to collect any data for that product.

Note that if you create your facility selections without specifying a version for the facilities, DECtrace always uses the latest version of the facility installed on your system. You won't have to change your facility selections when you perform an installation. However, you must recognize that any data collected from the new version may not be compatible with data collected before the installation. You would not be able to merge new data into an existing formatted database or file.

9.3.3 Installing a New Version of Rdb/VMS

Before installing a new version of Rdb/VMS, you *must* perform a full RMU[1] backup of the DECtrace administration and history databases (as well as any other Rdb/VMS databases on your system, including DECtrace-formatted databases produced with the FORMAT command).

To backup the DECtrace administration database, use the following command:

```
$  RMU/BACKUP EPC$ADMIN_DB EPC$ADMIN_DB.RBF
```

To backup the DECtrace history database, use the following command:

```
$  RMU/BACKUP EPC$HISTORY_DB EPC$HISTORY_DB.RBF
```

After installing the new version of Rdb/VMS you must restore your databases. The RMU/RESTORE operation restores the database to its original location.

To restore the DECtrace administration database, use the following command:

```
$  RMU/RESTORE/NEW_VERSION EPC$ADMIN_DB.RBF
```

To restore the DECtrace history database, use the following command:

```
$  RMU/RESTORE/NEW_VERSION EPC$HISTORY_DB.RBF
```

Because the DECtrace administration database is not available during the installation, Rdb/VMS inserts its facility definition into the DECtrace facility library SYS$SHARE:EPC$FACILITY.TLB. You must extract this definition and insert it into the DECtrace administration database. For example:

```
$  LIBRARY /TEXT /EXTRACT=RDBVMSV3.1-0 /OUT=RDBVMS.EPC$DEF -
_$  SYS$SHARE:EPC$FACILITY.TLB
$  COLLECT INSERT DEFINITION RDBVMS.EPC$DEF /REPLACE
```

Note *See the* VAX Rdb/VMS Release Notes *for the version number of your Rdb / VMS software.*

9.4 Managing the History Database

DECtrace maintains a user-visible history database which contains a record of all informational and error messages encountered during data collection. There is no automatic purging function and over time, the history database can become very large. You should periodically create a new (empty) database and either offload or delete the old version.

[1] RMU is the Rdb/VMS Management Utility

Use the SET HISTORY /NEW_FILE command to create a new version of the history database. For example:

```
$ COLLECT SET HISTORY /NEW_FILE
$ PURGE/LOG EPC$HISTORY_DB
%PURGE-I-FILPURG, EPC$DATABASE_DIR:EPC$HISTORY_DB.RDB;1 deleted
(22640 blocks)
```

One method of regularly creating a new history database is to put the DECtrace SET HISTORY/NEW_FILE command into your site-specific startup procedure. Each time your system reboots, the old history database is closed and a new one is created. If you choose to do this, you should put the command before the execution of EPC$STARTUP so that no history data is lost. For example:

```
$ COLLECT SET HISTORY/NEW_FILE
$ PURGE/NOLOG/NOCONFIRM/KEEP=2 -
    SYS$SYSDEVICE:[EPC.DATABASES]EPC$HISTORY_DB.RDB
$ @SYS$STARTUP:EPC$STARTUP
```

The DECtrace Registrar process binds to the history database once when it is first created, and unbinds when you issue either the STOP SYSTEM or the SET HISTORY/NEW_FILE command. The modify date of the history database is updated only when the Registrar unbinds. Because of this, incremental backups of your disks may not copy the history database (or the DECtrace administration database).

The solution is to create a new history database prior to performing your incremental backups. Alternately, you could perform an Rdb/VMS online backup using the RMU/BACKUP/ONLINE command.

9.5 Disabling Data Collection

You can assign the logical, EPC$DISABLED, to turn off process registration within the scope of the logical name. To specify the logical name table (which determines the scope) where you want to enter a logical name, use the /PROCESS, /JOB, /GROUP, /SYSTEM, or /TABLE qualifier. For example, the following command disables all process registrations in your UIC group:

```
$ ASSIGN/GROUP EPC$DISABLED EPC$DISABLED
```

See the ASSIGN command in the *VMS DCL Dictionary* for more information about creating logical names.

9.6 Improving DECtrace Performance

After you install DECtrace, you might want to adjust your system to enhance performance or lower the use of some system resources. One recommendation is to increase process working set parameters to speed up DECtrace formatting operations and report generations.

Another possible improvement relates to the formatting of large data collection files into Rdb/VMS databases. The logical name RDMS$BIND_WORK_FILE lets you specify temporary tables that Rdb/VMS uses on a disk structure other than the disk that contains the database, thus reducing the disk I/O operation on the disk where the database resides. The default location is SYS$LOGIN:. For example:

```
$ ! Assign the work area to another disk with read-write access
$ DEFINE RDMS$BIND_WORK_FILE WORK$DISK:[RDB.WORK]
```

9.7 Troubleshooting DECtrace

This section describes some typical problems that you might encounter when using DECtrace on your system. Table 9–2 shows the symptoms, causes, and cures of many common problems.

Note that the primary troubleshooting tool is the SHOW HISTORY command which displays error or warning messages that occur during data collection. See Section 3.5.2 for a description of the SHOW HISTORY command.

Table 9–2 Troubleshooting DECtrace Problems

Symptom	Possible Cause	Solution
Data collection is active but no data is collected.	No applications contained facilities that were specified in the facility selection.	Use SHOW REGISTER to show all available applications.
	Your output disk is full.	Purge old files and spread new data files across several disks using the /FILELIST qualifier to the SCHEDULE COLLECTION command.
	The DECtrace registrar has stopped.	Restart the registrar process. See Section 9.2.1.
Output disks fill up too quickly.	You are collecting too much data.	Use a collection class to limit the amount of data collected. See Section 2.1.

(continued on next page)

Table 9–2 (Cont.) Troubleshooting DECtrace Problems

Symptom	Possible Cause	Solution
	Insufficient disk space to begin with.	Spread data files over several disks using the /FILELIST qualifier to the SCHEDULE COLLECTION command.
	History database has grown too large.	Start a new history database with the SET HISTORY/NEW_FILE command and offload the old version.
You cannot delete facility definitions or facility selections or you cannot cancel collections.	You are not the creator of the specified definition, selection or collection.	You must have VMS BYPASS OR SYSPRV privilege to delete other users' definitions or selections.
	You cannot delete a facility definition that is referenced by a facility selection.	You must delete any facility selections that reference the facility definition.
	You cannot delete a selection that is referenced by an active or pending collection.	You must cancel any active or pending collections that reference the facility selection.
Collections only run on the local node (in a cluster environment).	You did not specify /CLUSTER on the SCHEDULE COLLECTION command.	Schedule a cluster-wide collection. See Section 3.2.2.
	DECtrace is not running on the rest of the nodes in the cluster.	You must start DECtrace on every node in the cluster. See Section 9.2.1.
	The EPC$ADMIN_DB and EPC$HISTORY_DB logicals do not reference the same databases on all nodes in the cluster.	Reinstall DECtrace. When prompted for EPC$ROOT, specify a directory that is common across all nodes in the cluster.

(continued on next page)

Table 9–2 (Cont.) Troubleshooting DECtrace Problems

Symptom	Possible Cause	Solution
Elapsed time information for duration events is negative.	The system time was changed during the execution of the event.	Reschedule data collection and do not change system time during active collections.
	The start event and end event service routines are switched in your facility.	Move the EPC$START_EVENT and EPC$END_EVENT service routine calls to their proper location and recompile your program.
You cannot stop the DECtrace registrar process (the STOP SYSTEM command fails).	The Rdb/VMS monitor process is not running.	Use the VMS STOP/ID command to stop the EPC$REGISTRAR process. See Section 9.2.2.

9.7.1 Error Messages and Recovery Procedures

The file SYS$HELP:EPC$MSG.DOC contains a listing of a subset of the DECtrace error messages, with explanations and user actions. The messages are arranged in alphabetical order by error code. You can use a text editor or the DCL SEARCH command to locate specific error codes or text strings.

Within the DECtrace command environment, you can use the HELP command to display the explanation and user action associated with an error message. For example, the following command displays information about the error code FACCRE_NOEVTOPTS:

```
DECtrace> HELP ERROR FACCRE_NOEVTOPTS

Error_Messages_and_Recovery

  FACCRE_NOEVTOPTS

    Must specify list of events or event options

       Explanation: User attempted to create a facility definition
       which did not contain any events.

       User Action: You must specify at least one event when creating
       a facility definition.
```

Note that an error encountered by a process which has registered with DECtrace is not related to any collection for which the process might be recording event data. If you use the COLLECTION-NAME parameter to the SHOW HISTORY command, you will not see errors or messages generated by the individual processes. For example, if a process is unable to record data due to a file protection violation on the data collection file, the message will be on the SHOW HISTORY /FORMAT=ALL report, not the report for a specific collection.

A Formatted Database and File Layouts

The DECtrace formatter component formats the DECtrace data collection files (DCFs) into either an Rdb/VMS database or a VAX RMS file. The formatted Rdb/VMS database or VAX RMS file can then be used for reporting and analyzing.

A.1 The Rdb/VMS Database

You can use the DECtrace formatter to merge, format, and store the event data collected from multiple collections that reference the same facility selection into an Rdb/VMS database.

Only collections of identically defined selections can be merged into a single database. Identically defined selections have identical data for the following fields:

- For the EPC$SELECTION relation—all fields except SELECTION_RECORD_ID (SELECTION_NAME and SELECTION_COMMENT)
- For the EPC$FACILITY relation—all fields except SELECTION_RECORD_ID (FACILITY_NUMBER, FACILITY_NAME, FACILITY_VERSION, FACILITY_DEFINITION_TIME, and COLLECTION_CLASS)
- For the EPC$EVENT relation—all fields except SELECTION_RECORD_ID and RELATION_NAME (FACILITY_NUMBER, EVENT_NUMBER, EVENT_NAME, EVENT_HEADER, EVENT_TYPE, and ITEM_FLAGS)
- For the EPC$ITEM relation—all fields except SELECTION_RECORD_ID (FACILITY_NUMBER, ITEM_NUMBER, ITEM_NAME, ITEM_HEADER, ITEM_TYPE, ITEM_WIDTH, ITEM_MAX_SIZE, ITEM_USAGE, and ITEM_CHARACTERISTICS)

- For the EPC$EVENT_ITEM relation—all fields except SELECTION_RECORD_ID (FACILITY_NUMBER, EVENT_NUMBER, ITEM_NUMBER, ITEM_POSITION, ITEM_USAGE_POINT, ITEM_USAGE_START, and ITEM_USAGE_END)

This version of the DECtrace formatter will only format and merge Version 1.0-0 of the DECtrace data collection file. The DCF_VERSION field within the data collection file should have the value "V1.0-0". It will only merge the data into an Rdb/VMS database created by Version 1.0-0 of the DECtrace formatter. The FORMAT_VERSION field of the EPC$IDENT relation in the Rdb/VMS database should also have the value "V1.0-0". Moreover, it will only merge those DCFs that have not been merged. When a DCF is merged, the information that uniquely identifies the file is stored in the EPC$DCF relation. The DECtrace formatter skips a DCF that is not mergeable.

A.1.1 The Data Types

Table A–1 shows the data types that are used in Rdb/VMS to represent the DECtrace-supported data types.

Table A–1 Rdb/VMS Representations of DECtrace-Supported Data Types

DECtrace-Supported Data Type	Rdb/VMS Data Type
BYTE	SMALLINT
WORD	SMALLINT
LONGWORD	INTEGER
QUADWORD	QUADWORD
ASCIC	VARCHAR of up to 255 characters
ASCIW	VARCHAR of up to 16383 characters
FIXED_ASCIC	VARCHAR of up to 255 characters

There are two types of relations in the database for storing two types of data:

1. Control information—collection information, facility information
2. Collected event data

The following sections describe in detail the relations in the database. Field names in **bold** signify key fields. Field names in *italic* signify foreign key fields.

Note *Some of the relations described here may be views that are defined over other base relations. The format of these base relations may change over time for improvements in performance and/or space utilization of the database. The* *names and formats of the relations / views and fields described here, however, are unlikely to change.*

A.1.2 String Data Segmentation

A form of string segmentation has been implemented to improve formatting performance as well as save storage space. It applies to all data items of type FIXED_ASCIC, ASCIC or ASCIW in all event data relations.

All collected string data of type FIXED_ASCIC, ASCIC, or ASCIW are stored in multiple segments, except when the entire string is small enough to fit in the first segment. Usually, the first segment of a string datum is stored in the base event data relation, while additional segments are stored in a separate relation for segmented strings only. There is one segmented-string relation for each base event data relation that has at least one data item of type FIXED_ASCIC, ASCIC, or ASCIW, in the event data relation. The storage space for the first segment could be specified to be of size zero, in which case all string segments for this data field are stored in the segmented-string relation.

All segments of a single datum are inter-related through a string ID that is unique within an event data relation. For every collected data item of type FIXED_ASCIC, ASCIC, or ASCIW, there is an additional field in the same event data relation for the string ID. The name of this additional field is <item_name>_STR_ID. For example, WHERE_BLOCK_STR_ID is the field for the string IDs of the data for data item WHERE_BLOCK of a point event, and WHERE_BLOCK_END_STR_ID is the field for the string IDs of the end data of a duration event. A value of 0 (zero) in this field indicates that there is only one string segment: the first, for a datum.

EPC$1_40_REQUEST is an example of an event data relation where two of the data item, ARGUMENT_LIST and WHERE_BLOCK, are of the ASCIW type, and one item, TARGET_DBKEY, is of the ASCIC type. Therefore, three additional fields, with a domain of STR_ID_DOMAIN (type INTEGER), are also created for the string IDs of these data fields. Table A–2 describes the fields in these relations.

Table A–2 The point event relation EPC$1_40_REQUEST

Field name	Rdb/VMS Data Type
COMP_STATUS	INTEGER
CODE_QUADWORD	QUADWORD
ARGUMENT_LIST	VARCHAR(x)
ARGUMENT_LIST_STR_ID	INTEGER

(continued on next page)

Table A-2 (Cont.) The point event relation EPC$1_40_REQUEST

Field name	Rdb/VMS Data Type
WHERE_BLOCK	VARCHAR(y)
WHERE_BLOCK_STR_ID	INTEGER
TARGET_DBKEY	VARCHAR(z)
TARGET_DBKEY_STR_ID	INTEGER

Segmented-string relations are named <event_data_relation_name>_ST, and have three fields. Table A-3 describes the fields in these relations.

Table A-3 A relation for segmented strings

Field name	Rdb/VMS Data Type	Description
STR_ID	INTEGER	The string ID of the datum.
SEGMENT_NUMBER	SMALLINT	Segment sequence number, starts with 1.
STR_SEGMENT	VARCHAR(n)	String segment.

One index will be created for each of these relations, on the STR_ID field. The index will be named the same as the relation itself.

Two data fields in two of the DECtrace system relations indicate the segment sizes of the segmented strings:

1 The ITEM_FIRST_SEGMENT_SIZE field, of domain ITEM_FIRST_SEGMENT_SIZE_DOMAIN, a SMALLINT, in the EPC$EVENT_ITEM relation, indicates the storage size for the first string segment of the data item in the base event data relation.

2 SEGMENT_SIZE, of domain EVENT_SEGMENT_SIZE_DOMAIN, a SMALLINT, in the EPC$EVENT relation, indicates the segment size of all the additional string segments in the segmented-string relation for the event.

A.1.3 Relations for the Control Information

There are 13 relations that hold control information: EPC$IDENT, EPC$SELECTION, EPC$COLLECTION, EPC$DCF, EPC$REG_PROCESS, EPC$FACILITY, EPC$EVENT, EPC$ITEM, EPC$EVENT_ITEM, EPC$PROCESS, EPC$IMAGE, EPC$DCF_IMAGE, and EPC$FACILITY_IMAGE.

A.1.3.1 The EPC$IDENT Relation

The EPC$IDENT relation holds information of the DECtrace collector that collected the data.

The IDENT_RECORD_ID field is a formatter-introduced field used as the key field and is indexed. Table A–4 describes the fields in the EPC$IDENT relation.

Table A–4 EPC$IDENT Relation

Field Name	Data Type/Size	Description
IDENT_RECORD_ID	Signed word	The EPC$IDENT record identifier.
PRODUCT_VERSION	Varying string, 10 bytes	The DECtrace product version string.
DCF_VERSION	Varying string, 10 bytes	The data collection file version string.
FORMAT_VERSION	Varying string, 10 bytes	The formatted database version string.

A.1.3.2 The EPC$SELECTION Relation

The EPC$SELECTION relation holds the facility selection definition information.

The SELECTION_RECORD_ID field is a formatter-introduced field used as the key field and is indexed. Table A–5 describes the fields in the EPC$SELECTION relation.

Table A–5 EPC$SELECTION Relation

Field Name	Data Type/Size	Description
SELECTION_RECORD_ID	Signed word	The EPC$SELECTION record identifier. Range: 1 to 999 inclusive.
SELECTION_NAME	Varying string, 32 bytes	The selection name.
SELECTION_COMMENT	Varying string, 80 bytes	The comment for the selection.

A.1.3.3 The EPC$COLLECTION Relation

The EPC$COLLECTION relation holds the collection information.

The COLLECTION_RECORD_ID field is a formatter-introduced field used as the key field and is indexed. The SELECTION_RECORD_ID field serves as a foreign key from relation EPC$SELECTION. Table A–6 describes the fields in the EPC$COLLECTION relation.

Table A–6 EPC$COLLECTION Relation

Field Name	Data Type/Size	Description
COLLECTION_RECORD_ID	Signed word	The EPC$COLLECTION record identifier.
COLLECTION_NAME	Varying string, 32 bytes	The collection name.
START_TIME	Date	The collection start time.
END_TIME	Date	The collection end time.
SELECTION_RECORD_ID	Signed word	The EPC$SELECTION record identifier.

A.1.3.4 The EPC$DCF Relation The EPC$DCF relation holds the data collection file information.

The DCF_RECORD_ID field is a formatter-introduced field used as the key field and is indexed. The COLLECTION_RECORD_ID field serves as a foreign key from relation EPC$COLLECTION. Table A–7 describes the fields in the EPC$DCFzex relation.

Table A–7 EPC$DCF Relation

Field Name	Data Type/Size	Description
DCF_RECORD_ID	Signed word	The EPC$DCF record identifier.
SYSTEM_ID	Integer, 6 bytes	The system identification of the node, obtain with an SYS$GETSYI call, specifying SYI$_NODE_SYSTEMID.
ACTIVATION_TIME	Date	Collection activation time.
TOTAL_FILES	Signed word	The total number of DCFs from the collection.
FILE_NUMBER	Signed word	The file number of this DCF.
FILE_NAME	Varying string, 255 bytes	The file name of this DCF.
COLLECTION_RECORD_ID	Signed word	The EPC$COLLECTION record identifier.

A.1.3.5 The EPC$REG Relation The EPC$REG relation holds the registration identification information.

The COLLECTION_RECORD_ID field serves as a foreign key from relation EPC$COLLECTION.

Table A–8 describes the fields in the EPC$REG relation.

Table A–8 EPC$REG Relation

Field Name	Data Type/Size	Description
COLLECTION_RECORD_ID	Signed word	The EPC$COLLECTION record identifier.
REG_ID	Varying string, 255 bytes	The registration ID.

A.1.3.6 The EPC$FACILITY Relation The EPC$FACILITY relation holds the facility definition information.

The SELECTION_RECORD_ID and FACILITY_NUMBER together form the key field and are indexed. The SELECTION_RECORD_ID field also serves as a foreign key from relation EPC$SELECTION. Table A–9 describes the fields in the EPC$FACILITY relation.

Table A–9 EPC$FACILITY Relation

Field Name	Data Type/Size	Description
SELECTION_RECORD_ID	Signed word	The EPC$SELECTION record identifier.
FACILITY_NUMBER	Signed word	The facility number.
FACILITY_NAME	Varying string, 27 bytes	The facility name.
FACILITY_VERSION	Varying string, 10 bytes	The facility version.
FACILITY_DEFINITION_TIME	Date	Facility definition creation date.
COLLECTION_CLASS	Varying string, 32 bytes	The collection class name.

A.1.3.7 The EPC$EVENT Relation The EPC$EVENT relation holds the event information for a facility.

The SELECTION_RECORD_ID, FACILITY_NUMBER and EVENT_NUMBER together form the key field for this relation and are indexed. The SELECTION_RECORD_ID field also serves as a foreign key from relation EPC$SELECTION. The FACILITY_NUMBER field also serves as a foreign key from relation EPC$FACILITY. Table A–10 describes the fields in the EPC$EVENT Relation.

Table A–10 EPC$EVENT Relation

Field Name	Data Type/Size	Description
SELECTION_RECORD_ID	Signed word	The EPC$SELECTION record identifier.
FACILITY_NUMBER	Signed word	The facility each event is associated with.
EVENT_NUMBER	Signed word	The event number for this event.
EVENT_NAME	Varying string, 15 bytes	Name describing the event.
EVENT_HEADER	Varying string, 15 bytes	The header used in reports.
EVENT_TYPE	Signed word	The event type. Possible values follow: ▪ EPC$K_EVENT_TYPE_POINT(81)—Indicates a point event type. ▪ EPC$K_EVENT_TYPE_DURATION(84)—Indicates a duration event type.
ITEM_FLAGS	Bit array, 16 bytes	The item flags for the event.
SEGMENT_SIZE	Signed word	Size of the segmented string.
RELATION_NAME	Varying string, 28 bytes	The name of the relation that holds the captured data for this event.

A.1.3.8 The EPC$ITEM Relation The EPC$ITEM relation holds the item information for a facility.

The SELECTION_RECORD_ID, FACILITY_NUMBER, and ITEM_NUMBER together form the key field for this relation and are indexed. The SELECTION_RECORD_ID field also serves as a foreign key from relation EPC$SELECTION. The FACILITY_NUMBER field also serves as a foreign key from relation EPC$FACILITY. Table A–11 describes the fields in the EPC$ITEM relation.

Table A–11 EPC$ITEM Relation

Field Name	Data Type/Size	Description
SELECTION_RECORD_ID	Signed word	The EPC$SELECTION record identifier.
FACILITY_NUMBER	Signed word	The facility each item is associated with.
ITEM_NUMBER	Signed word	The item number for this event.
ITEM_NAME	Varying string, 15 bytes	Name describing the item.
ITEM_HEADER	Varying string, 15 bytes	The column header used in reports.
ITEM_TYPE	Signed word	The data type of the item.
ITEM_RADIX	Signed word	The radix of the item.
ITEM_WIDTH	Signed word	The column width used in reports.
ITEM_MAX_SIZE	Signed word	The maximum size of the item.
ITEM_USAGE	Signed word	Usage of the item. Possible values follow: ■ EPC$K_ITM_U_LEVEL(1)—Indicates the item is a level. A level is a meter or gauge that indicates the "CURRENT" value of some metric. Its value may go up & down. ■ EPC$K_ITM_U_COUNTER(2)—Indicates the item is a counter. A counter indicates the number of times something occurred. Its value increases over time. ■ EPC$K_ITM_U_PERCENT(3)—Indicates the item is a percentage. A percentage is a type of level. ■ EPC$K_ITM_U_TEXT(4)—Indicates the item is text. ■ EPC$K_ITM_U_PRIVATE(5)—Indicates the item is for private use by the facility.

(continued on next page)

Table A–11 (Cont.) EPC$ITEM Relation

Field Name	Data Type/Size	Description
ITEM_CHARACTERISTICS	Signed longword	Item value characteristics. Possible values follow: ■ EPC$K_ITM_CHR_PRINT(1)— Printable item value. ■ EPC$K_ITM_CHR_NONPRINT(2)— Non-printable item value.

A.1.3.9 The EPC$EVENT_ITEM Record

The EPC$EVENT_ITEM relation holds the relationship information between the events and items of a facility.

The SELECTION_RECORD_ID, FACILITY_NUMBER, EVENT_NUMBER, and ITEM_NUMBER together form the key field for this relation and are indexed. The SELECTION_RECORD_ID field also serves as a foreign key from relation EPC$SELECTION. The FACILITY_NUMBER field also serves as a foreign key from relation EPC$FACILITY. The EVENT_NUMBER field also serves as a foreign key from relation EPC$EVENT. The ITEM_NUMBER field also serves as a foreign key from relation EPC$ITEM. Table A–12 describes the fields in the EPC$EVENT relation.

Table A–12 EPC$EVENT_ITEM Relation

Field Name	Data Type/Size	Description
SELECTION_RECORD_ID	Signed word	The EPC$SELECTION record identifier.
FACILITY_NUMBER	Signed word	The facility each event-item pair is associated with.
EVENT_NUMBER	Signed word	The event number of the event-item pair.
ITEM_NUMBER	Signed word	The item number of the event-item pair.
ITEM_POSITION	Signed word	The item position in the event record.

(continued on next page)

Table A–12 (Cont.) EPC$EVENT_ITEM Relation

Field Name	Data Type/Size	Description
ITEM_FIRST_SEGMENT_SIZE	Signed word	The size the first segment of the segmented string.
ITEM_USAGE_START	Signed word	A value of one indicates that this data item will be collected in a START event, for the above event number. A value of zero indicates otherwise.
ITEM_USAGE_END	Signed word	A value of one indicates that this data item will be collected in an END event, for the above event number. A value of zero indicates otherwise.

A.1.3.10 The EPC$PROCESS Relation

The EPC$PROCESS relation holds the process information.

The PROCESS_RECORD_ID field is a formatter-introduced field used as the key field and is indexed. The NODE field is also indexed. The IDENT_RECORD_ID field serve as a foreign key from relation EPC$IDENT. Table A–13 describes the fields in the EPC$PROCESS relation.

Table A–13 EPC$PROCESS Relation

Field Name	Data Type/Size	Description
PROCESS_RECORD_ID	Signed word	The EPC$PROCESS record identifier.
EPID	Signed longword	The extended process identity number.
USERNAME	Varying string, 12 bytes	The username of the process.
ACCOUNT	Varying string, 8 bytes	The process account name.
UIC	Signed longword	The process user identification.
PROCESS_NAME	Varying string, 15 bytes	The process name.
CREATION_TIME	Date	Process creation time.
BASE_PRIORITY	Signed longword	The base priority of the process.
PROCESS_MODE	Signed longword	The process mode.

(continued on next page)

Table A–13 (Cont.) EPC$PROCESS Relation

Field Name	Data Type/Size	Description
SYSTEM_ID	Integer, 6 bytes	The system identification of the node, obtained with a SYS$GETSYI call, specifying SYI$_NODE_SYSTEMID.
NODE_NAME	Varying string, up to 404 bytes	The full DECNET node name, obtained with a SYS$GETSYI call, specifying SYI$_NODENAME.
NODE	Varying string, up to 64 bytes	The abbreviated version of the above DECNET node name.
CPU_TYPE	Varying string, 31 bytes	The CPU type, obtained with a SYS$GETSYI call, specifying SYI$_HW_NAME.
VMS_VERSION	Varying string, 8 bytes	The VMS version, obtained with a SYS$GETSYI call, specifying SYI$_VERSION.
BOOT_TIME	Date	The boot time of the node, result of a SYS$GETSYI call when specifying SYI$_BOOTTIME.
IDENT_RECORD_ID	Signed word	The EPC$IDENT record identifier.

A.1.3.11 The EPC$IMAGE Relation

The EPC$IMAGE relation holds the image information.

The IMAGE_RECORD_ID field is a formatter-introduced field used as the key field and is indexed. The PROCESS_RECORD_ID field serves as a foreign key from relation EPC$PROCESS. Table A–14 describes the fields in the EPC$IMAGE relation.

Table A–14 EPC$IMAGE Relation

Field Name	Data Type/Size	Description
IMAGE_RECORD_ID	Signed longword	The EPC$IMAGE record identifier.
IMAGE_NAME	Varying string, 255 bytes	The full file specification of the image.
ACTIVATION_TIME	Date	Image activation time.

(continued on next page)

Table A–14 (Cont.) EPC$IMAGE Relation

Field Name	Data Type/Size	Description
LINK_TIME	Date	Image link time.
IMAGE_ID	Varying string, 15 bytes	The image identity number.
PROCESS_RECORD_ID	Signed word	The EPC$PROCESS record identifier.

A.1.3.12 The EPC$DCF_IMAGE Relation The EPC$DCF_IMAGE relation captures the relationship information between data collection files and execution images.

The DCF_RECORD_ID and IMAGE_RECORD_ID together form the key field for this relation and are indexed. The DCF_RECORD_ID field also serves as a foreign key from relation EPC$DCF. The IMAGE_RECORD_ID field also serves as a foreign key from relation EPC$IMAGE. Table A–15 describes the fields in the EPC$DCF_IMAGE relation.

Table A–15 EPC$DCF_IMAGE Relation

Field Name	Data Type/Size	Description
DCF_RECORD_ID	Signed word	The EPC$DCF record identifier.
IMAGE_RECORD_ID	Signed longword	The EPC$IMAGE record identifier.

A.1.3.13 The EPC$FACILITY_IMAGE Relation The EPC$FACILITY_IMAGE relation holds the relationship information between facilities and execution images.

The SELECTION_RECORD_ID, FACILITY_NUMBER, and IMAGE_RECORD_ID together form the key field for this relation and are indexed. The SELECTION_RECORD_ID field also serves as a foreign key from relation EPC$SELECTION. The FACILITY_NUMBER field also serves as a foreign key from relation EPC$FACILITY. The IMAGE_RECORD_ID field also serves as a foreign key from relation EPC$IMAGE. Table A–16 describes the fields in the EPC$FACILITY_IMAGE relation.

Table A–16 EPC$FACILITY_IMAGE Relation

Field Name	Data Type/Size	Description
SELECTION_RECORD_ID	Signed word	The EPC$SELECTION record identifier.
FACILITY_NUMBER	Signed word	The facility number.
FACILITY_VERSION	Varying string, 10 bytes	The facility version.
IMAGE_RECORD_ID	Signed longword	The EPC$IMAGE record identifier.
REG_ID	Varying string, 255 bytes	The facility registration ID.

A.1.4 Relations for the Collected Data

A relation for the data collected for an event is created only if data has been collected for that event. The name for an event data relation is: EPC$<SELECTION_RECORD_ID>_<FACILITY_NUMBER>_<EVENT_NAME>.

For a point event data relation, the item name of an item is used as the column name in the relation.

For a duration event data relation, the suffixes _START and _END are added to the item names for items that are collected at event start time and event end time respectively.

For example, an item named MEMORY_USAGE will have a column name MEMORY_USAGE in a point event data relation. A similar item will have a column name MEMORY_USAGE_START for item data collected at event start time. A similar item will have a column name MEMORY_USAGE_END for item data collected at event end time.

In addition to facility-defined events and process items, all event relations will also include the fields described in Table A–17.

Table A–17 Event Relation Fields

Field Name	Data Type/Size	Description
COLLECTION_RECORD_ID	Signed word	The EPC$COLLECTION record identifier, a foreign key.
IMAGE_RECORD_ID	Signed longword	The EPC$IMAGE record identifier, a foreign key.
CONTEXT_NUMBER	Signed longword	The context number.
TIMESTAMP_POINT	Date	Point event logged time, for point events only.
TIMESTAMP_START	Date	Duration event start time, for duration events only.
TIMESTAMP_END	Date	Duration event end time, for duration events only.

A.1.5 The Entity Relationship Diagram for the Database

Figure A–1 shows the ER diagram for the formatted database.

A.1.6 The Predefined Views

To facilitate data retrieval, the following views are predefined:

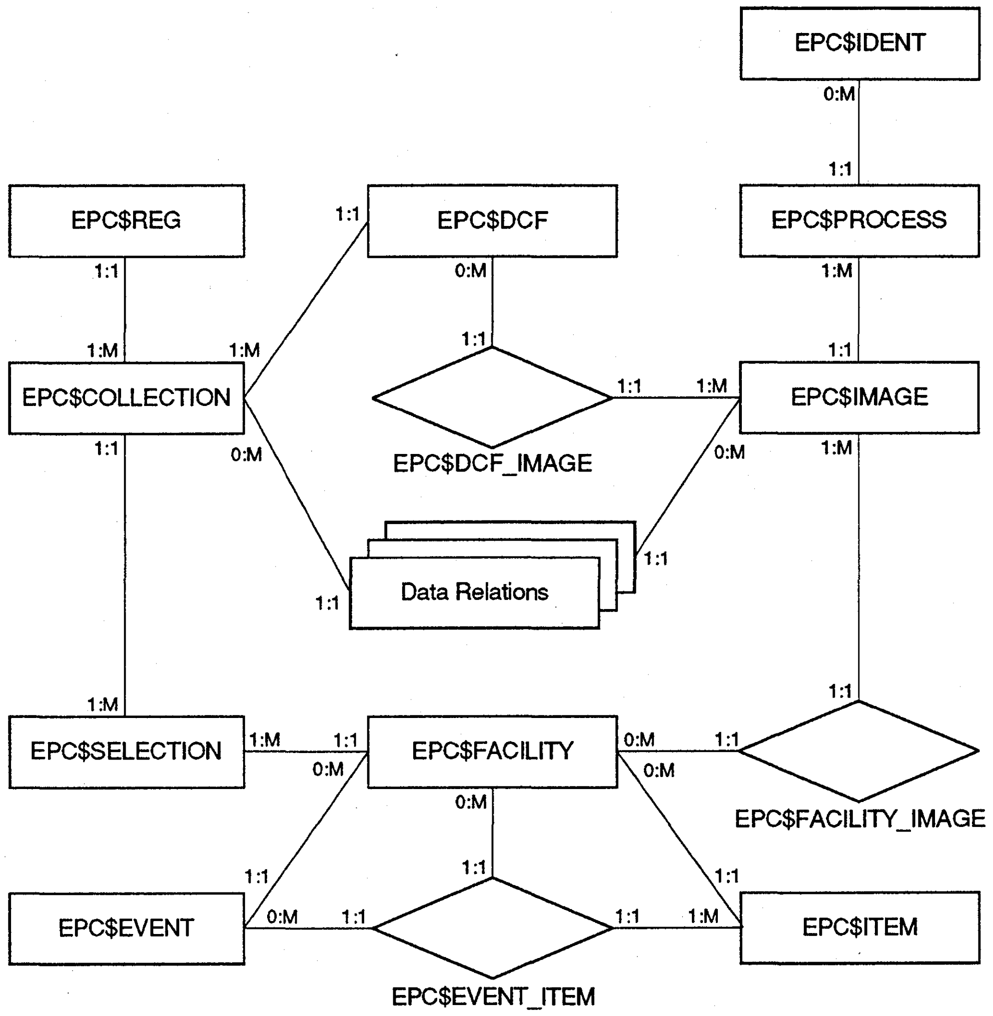
A.1.6.1 Relating Collection Information with Event Data A view is defined for each event relation to relate collection information to the events:

```
CREATE VIEW <Event RELATION_NAME>_C
AS
SELECT *
FROM   <Event RELATION_NAME>, EPC$COLLECTION
WHERE  <Event RELATION_NAME>.COLLECTION_RECORD_ID =
       EPC$COLLECTION.COLLECTION_RECORD_ID
```

A.1.6.2 Relating Selection and Collection Information with Event Data A view is defined for each event relation to relate selection and collection information to the events:

```
CREATE VIEW <Event RELATION_NAME>_SC
AS
SELECT *
FROM   <Event RELATION_NAME>, EPC$COLLECTION, EPC$SELECTION
WHERE  <Event RELATION_NAME>.COLLECTION_RECORD_ID =
       EPC$COLLECTION.COLLECTION_RECORD_ID
  AND  EPC$COLLECTION.SELECTION_RECORD_ID = EPC$SELECTION.SELECTION_RECORD_ID
```

Figure A–1 The Entity Relationship Diagram for the Database



NU–2054A–RA

A.1.6.3 Relating Image Information with Event Data A view is defined for each event relation to relate image information to the events:

```
CREATE VIEW <Event RELATION_NAME>_I
AS
SELECT *
FROM   <Event RELATION_NAME>, EPC$IMAGE
WHERE  <Event RELATION_NAME>.IMAGE_RECORD_ID = EPC$IMAGE.IMAGE_RECORD_ID
```

A.1.6.4 Relating Process and Image Information with Event Data

A view is defined for each event relation to relate process and image information to the events:

```
CREATE VIEW <Event RELATION_NAME>_PI
AS
SELECT *
FROM   <Event RELATION_NAME>, EPC$IMAGE, EPC$PROCESS
WHERE  <Event RELATION_NAME>.IMAGE_RECORD_ID = EPC$IMAGE.IMAGE_RECORD_ID
  AND  EPC$IMAGE.PROCESS_RECORD_ID = EPC$PROCESS.PROCESS_RECORD_ID
```

A.1.7 Limitations

Due to the range limitation placed on SELECTION_RECORD_ID and the sizes of COLLECTION_RECORD_ID, IDENT_RECORD_ID, PROCESS_RECORD_ID, and IMAGE_RECORD_ID, the maximum numbers of unique record that can be stored in EPC$SELECTION, EPC$COLLECTION, EPCDCF, EPCIDENT, EPC$PROCESS, and EPC$IMAGE relations are 999, 32768, 32768, 32768, 32768, and 2147483647 respectively.

A.2 The VAX RMS File Format

This section describes the formatted VAX RMS file created by DECtrace when you specify the /TYPE=RMS qualifier to the FORMAT command.

The DECtrace formatter component merges, formats, and stores the event data collected from multiple collections of the same facility selection into a formatted VAX RMS file. Formatted VAX RMS files are sequential files with variable length records. Maximum size of a record is 32,765 bytes. This is the maximum record size for a VAX RMS sequential file written to disk.

Only collections with identical facility selections can be merged into a single VAX RMS file. Identically defined selections have the same data for the following fields:

- For the COLLECTION record—SELECTION NAME and SELECTION COMMENT
- For the FACILITY record—FACILITY NUMBER, FACILITY NAME, FACILITY VERSION, FACILITY DEFINITION TIME, and COLLECTION CLASS
- For the EVENT record—FACILITY NUMBER, EVENT NUMBER, EVENT NAME, EVENT HEADER, EVENT TYPE, RESOURCE ITEM FLAGS, and everything in each COLLECTION ITEM DESCRIPTOR

This version of the DECtrace formatter will only format and merge Version 1.0-0 of the DECtrace data collection file. The DCF_VERSION field within the data collection file should have the value "V1.0-0". It only merges the data into a VAX RMS file created by Version 1.0-0 of the DECtrace formatter. The FORMAT_VERSION field of the IMAGE record in the VAX RMS file should also have the value "V1.0-0". Moreover, it only merges those DCFs that have not been merged. When a DCF is merged, the information that uniquely identifies the file is stored in the dictionary as well as the DCF record. The DECtrace formatter skips a DCF that is not mergeable.

A.2.1 Data Types

The following sections describe the data types used in the formatted VAX RMS file.

A.2.1.1 Date A date field contains a time and date in the standard VMS quadword binary format.

A.2.1.2 Fixed ASCIC/ASCIW String A fixed ASCIC string field records the actual length of the string in the first byte, followed by the actual string itself. A fixed ASCIW string field records the actual length of the string in the first word, followed by the actual string itself. The field size is fixed. The content of the additional space following the actual string is undetermined. For example, the SELECTION NAME field of the COLLECTION record is a field of 33 bytes long. The first byte of this field contains a count of the selection name, followed by the selection name of up to 32 characters. Therefore, for a selection name that is only five character long, the first byte of this field has a value of five, followed by the selection name, followed by 27 unused bytes with unpredictable values in them.

A.2.2 Records Organization

There are ten types of records: seven for control information (FMTDICT, COLLECTION, FACILITY, DCF, IMAGE, FACILITY-REGISTRATION, and EVENT records) and three for collected data (POINT, START, and END records). The first byte of a record contains a literal which indicates the type of the record. Table A–18 shows the possible record types.

Table A–18 Record Type Literal

Record Type	Literal
FMTDICT	EPC$K_FMTDICT_REC
COLLECTION	EPC$K_COLLECTION_REC
FACILITY	EPC$K_FACILITY_REC
DCF	EPC$K_DCF_REC
IMAGE	EPC$K_IMAGE_REC
FACILITY_REGISTRATION	EPC$K_FACREG_REC

(continued on next page)

Table A-18 (Cont.) Record Type Literal

Record Type	Literal
EVENT	EPC$K_EVENT_REC
POINT	EPC$K_POINT_REC
START	EPC$K_START_REC
END	EPC$K_END_REC

The formatter dictionary record (FMTDICT) contains dictionary information that is used by the formatter only. Instances of this record may appear anywhere in a formatted VAX RMS file. This record should simply be ignored and skipped over by any program that reads the VAX RMS file.

A formatted VAX RMS file consists of one or more collection sections. Each collection section has a COLLECTION record, followed by one or more FACILITY records and a collected data section. The appearance of a new COLLECTION record in a file signifies the end of the last collection section and the beginning of the next, and the information in the new COLLECTION record supersedes that of the previous one.

A collected data section is composed of one or more merged data collection files (DCFs). A DCF record signifies the end of the previous DCF and the beginning of a new one. Within a collected data section are a number of IMAGE records, FACILITY-REGISTRATION records, EVENT records, and the Data Collection records. An IMAGE record that describes an image precedes the data that was collected from that image. A FACILITY-REGISTRATION record indicates the registration of a facility through an image. The EPID in this record points to the last IMAGE record that has the same EPID. An EVENT record that describes an event precedes the first collected data record for that event. The last IMAGE record that has an EPID that matches the EPID field of a collected data record shows the image where the data was collected.

Note that data collected from the same collection may be separated by collection sections of some other collections, depending on the order in which the DCFs were merged. In this case, each of the separated parts is a complete collection section with an identical COLLECTION record, and most likely the same set of FACILITY and IMAGE records. They would have different DCF records, however.

A.2.2.1 Control Records The control records are described in the following sections.

A.2.2.1.1 FMTDICT Record

Table A–19 describes the fields in the formatter dictionary record.

Table A–19 FMTDICT Record

Field Name	Data Type	Bytes	Description
RECORD TYPE	Signed byte	1	Contains EPC$K_FMTDICT_REC to indicate that this is a formatter dictionary record.
DICTIONARY SIZE	Signed longword	4	The size of the dictionary to follow.
DICTIONARY	N bytes		The formatter dictionary.

A.2.2.1.2 COLLECTION Record

Table A–20 describes the fields in the COLLECTION record.

Table A–20 COLLECTION Record

Field Name	Data Type	Bytes	Description
RECORD TYPE	Signed byte	1	Contains EPC$K_COLLECTION_REC to indicate that this is a COLLECTION record.
SELECTION NAME	Fixed ASCIC string	33	The selection name.
SELECTION COMMENT	Fixed ASCIC string	81	The comment for the selection.
COLLECTION NAME	Fixed ASCIC string	33	The collection name.
START TIME	Date	8	The collection start time and date.
END TIME	Date	8	The collection end time and date.
REG ID LIST COUNT	Signed longword	4	Registration ID list count.
REG ID	Fixed ASCIC string	256	The registration ID.

A.2.2.1.3 FACILITY Record

Table A–21 describes the fields in the FACILITY record.

Table A–21 FACILITY Record

Field Name	Data Type	Bytes	Description
RECORD TYPE	Signed byte	1	Contains EPC$K_FACILITY_REC to indicate that this is a FACILITY record.
FACILITY NUMBER	Signed word	2	The facility number.
FACILITY NAME	Fixed ASCIC string	28	The facility name.
FACILITY VERSION	Fixed ASCIC string	11	The facility version.
FACILITY DEFINITION TIME	Date	8	Facility definition creation time.
COLLECTION CLASS	Fixed ASCIC string	33	The collection class name.

A.2.2.1.4 DCF Record Table A–22 describes the fields in the data collection file record.

Table A–22 DCF Record

Field Name	Data Type	Bytes	Description
RECORD TYPE	Signed byte	1	Contains EPC$K_DCF_REC to indicate that this is a Data Collection File information record.
ACTIVATION TIME	Date	8	Collection activation time.
SYSTEM ID	Integer	6	The system identification of the node, obtained with a SYS$GETSYI call, specifying SYI$_NODE_SYSTEMID.
TOTAL FILES	Signed word	2	The total number of DCFs from the collection.
FILE NUMBER	Signed word	2	The file number of this DCF.
FILE NAME	Fixed ASCIC string	256	The file name of this DCF.

A.2.2.1.5 IMAGE Record Table A–23 describes the fields in the IMAGE record.

Table A–23 IMAGE Record

Field Name	Data Type	Bytes	Description
RECORD TYPE	Signed byte	1	Contains EPC$K_IMAGE_REC to indicate that this is an IMAGE record.
PRODUCT VERSION	Fixed ASCIC string	11	The DECtrace product version string.
DCF VERSION	Fixed ASCIC string	11	The data collection file version string.
FORMAT VERSION	Fixed ASCIC string	11	The formatted VAX RMS file version string.
EPID	Signed longword	4	The extended process identity number.
UIC	Signed longword	4	The process user identification.
ACCOUNT	Fixed ASCIC string	9	The process account name.
USERNAME	Fixed ASCIC string	13	The username of the process.
PROCESS NAME	Fixed ASCIC string	16	The process name.
PROCESS CREATION TIME	Date	8	Process creation time.
BASE PRIORITY	Signed longword	4	The base priority of the process.
PROCESS MODE	Signed longword	4	The process mode.
SYSTEM ID	Integer	6	The system identification of the node, obtained with a SYS$GETSYI call, specifying SYI$_NODE_SYSTEMID.
NODE NAME	Fixed ASCIW string	406	The DECNET node name, obtained with a SYS$GETSYI call, specifying SYI$_NODENAME.
CPU TYPE	Fixed ASCIC string	32	The CPU type, obtained with a SYS$GETSYI call, specifying SYI$_HW_NAME.

(continued on next page)

Table A–23 (Cont.) IMAGE Record

Field Name	Data Type	Bytes	Description
VMS VERSION	Fixed ASCIC string	9	The VMS version, obtained with a SYS$GETSYI call, specifying SYI$_VERSION.
SYSTEM BOOT TIME	Date	8	The time the node was booted, result of an SYS$GETSYI call when specifying SYI$_ BOOTTIME.
IMAGE LINK TIME	Date	8	The time when this image was linked.
IMAGE ID	Fixed ASCIC string	16	The image identity number.
IMAGE NAME	Fixed ASCIC string	256	The full file specification of the image.
IMAGE ACTIVATION TIME	Date	8	Image activation time.

A.2.2.1.6 FACILITY-REGISTRATION Record Table A–24 describes the fields in the FACILITY-REGISTRATION record.

Table A–24 FACILITY-REGISTRATION Record

Field Name	Data Type	Bytes	Description
RECORD TYPE	Signed byte	1	Contains EPC$K_FACREG_ REC to indicate that this is an IMAGE record.
EPID	Signed longword	4	The extended process identity number.
FACILITY NUMBER	Signed word	2	The facility number.
FACILITY VERSION	Fixed ASCIC string	11	The facility version.
REG ID	Fixed ASCIC string	256	The registration ID.

A.2.2.1.7 EVENT Record Table A–25 describes the fields in the EVENT record.

Table A–25 Event Record Description

Field Name	Data Type	Bytes	Description
RECORD TYPE	Signed byte	1	Contains EPC$K_EVENT_REC to indicate that this is an EVENT record.
FACILITY NUMBER	Signed word	2	The facility each event is associated with.
EVENT NUMBER	Signed byte	1	The event number for this event.
EVENT NAME	Fixed ASCIC string	16	Name describing the event.
EVENT HEADER	Fixed ASCIC string	16	The column header used in reports.
EVENT TYPE	Signed byte	1	The event type. Possible values are: ■ EPC$K_EVENT_TYPE_POINT(81)—Indicates a point event type. ■ EPC$K_EVENT_TYPE_START(82)—Indicates a duration event type, with start items in the ITEM DESCRIPTOR LIST. ■ EPC$K_EVENT_TYPE_END(83)—Indicates a duration event type, with end items in the ITEM DESCRIPTOR LIST.
ITEM DESCRIPTOR LIST COUNT	Signed longword	4	Item descriptor list count.
ITEM DESCRIPTOR LIST	44n bytes		A list of item descriptor structures. See Section A.2.2.1.8.

A.2.2.1.8 Item Descriptor Table A–26 describes the fields in the item descriptor record.

Table A–26 Item Descriptor

Field Name	Data Type	Bytes	Description
ITEM NUMBER	Signed byte	1	The item number for this item.
ITEM NAME	Fixed ASCIC string	16	Name describing the item.
ITEM HEADER	Fixed ASCIC string	16	The column header used in reports.
ITEM TYPE	Signed word	2	The data type of the item.
ITEM WIDTH	Signed word	2	The column width used in reports.
ITEM MAX SIZE	Signed word	2	The maximum size of the item.
ITEM USAGE	Signed byte	1	Usage of the item. Possible values follow: ■ EPC$K_ITM_U_LEVEL(1)—Indicates the item is a level. A level is a meter or gauge that indicates the "CURRENT" value of some metric. Its value may go up and down. ■ EPC$K_ITM_U_COUNTER(2)—Indicates the item is a counter. A counter indicates the number of times something occurred. Its value is ever increasing. ■ EPC$K_ITM_U_PERCENT(3)—Indicates the item is a percentage. A percentage is a type of level. ■ EPC$K_ITM_U_TEXT(4)—Indicates the item is text. ■ EPC$K_ITM_U_PRIVATE(5)—Indicates the item is for private use by the facility.
ITEM CHARACTERISTICS	Signed longword	4	Item value characteristics. Possible values follow: ■ EPC$K_ITM_CHR_PRINT(1)—Printable item value. ■ EPC$K_ITM_CHR_NONPRINT(2)—Non-printable item value.

A.2.2.2 Data Collection Records The data collection records contain a fixed length portion with fields that point to the image and facility that generated the data. This is followed by the actual collected data. The optional process data items appears first, followed by the facility data items in the order the collection item descriptors appear in each EVENT record. Each data item occupies only as much space as needed according to the data type: four bytes for a longword, nine bytes for a ASCIC type with a count of eight in the count field and so forth.

Table A–27 describes the fields in the data collection record.

Table A–27 Data Collection Record

Field Name	Data Type	Bytes	Description
RECORD TYPE	Signed byte	1	Contains one of the following to indicate one of the data collection records: ■ EPC$K_POINT_REC—Indicates a point data collection record. ■ EPC$K_START_REC—Indicates a start data collection record. ■ EPC$K_END_REC—Indicates an end data collection record.
EPID	Signed longword	4	The extended process identity number.
TIME	Date	8	The record log time and date.
FACILITY NUMBER	Signed word	2	The facility that logged the event.
EVENT NUMBER	Signed byte	1	The event number of the logged event.
CONTEXT NUMBER	Signed longword	4	Context number.
HANDLE	Signed longword	4	Instance handle used to match start and end records.
COLLECTED DATA	Varied structure	n	The collected data.

Glossary

administration database

A single VAX Rdb/VMS database that contains the DECtrace facility definitions, facility selections, schedule information, and information about the registered and current VMS processes that are performing data collection on a given system. There is one DECtrace administration database per system or VAXcluster.

application program

A sequence of instructions and routines, not part of the basic operating system, designed to serve the specific needs of a user. An application program can be instrumented with DECtrace service routine calls. Also referred to as a **facility**, especially after the DECtrace calls have been added.

AST

See **asynchronous system traps**.

asynchronous system traps

A software-simulated interrupt to a user-defined service routine. ASTs enable a user process to be notified asynchronously, with respect to the process, of the occurrence of a specific event. Many of the DECtrace service routines use ASTs to reduce execution time.

collection class

A set (or group) of events and items that can be collected for a facility. Classes for a facility are specified in the facility definition. Users refer to a class when creating a facility selection. There can be one or more classes for each facility. Typical classes include CAPACITY_PLANNING, PERFORMANCE, and WORKLOAD.

collection interval

See **interval**

collection name

A 1- to 16-character string that represents the name of a particular collection.

data collection file

A file that contains raw data gathered during data collection. A single data collection file stores data for one or more VMS processes. The file is created and written to by the DECtrace service routines.

data collection

The process of collecting data on a system or VAXcluster. Criteria in the scheduling of data collection include when to collect data, where to put the output, and which facility selection to use. Data collection must be scheduled on a system in order to collect data. Only one data collection can actively collect data at any time on a given node.

data formatting

Organizing collected data into a VAX Rdb/VMS database or a formatted VAX RMS file. Collected data must be formatted before DECtrace can generate reports based on it.

data merging

Combining several data files into a single formatted database during data formatting.

DECtrace service routines

See **service routines**

duration event

See **event**

end event

The end of a duration event. See **event**

event

An occurrence of some activity within a facility. There are two types of events: **duration** and **point**. Duration events have logical beginning and ending points. Point events occur instantaneously. You can define up to 128 events for each facility.

event flag

A Boolean value corresponding to a particular event for a facility. If an event flag is set, it indicates that data capture is enabled for that event.

event flag list

A 128-element array of event flags for all events within a facility. Each facility has its own event flag list.

event handle

A unique numeric identifier generated by DECtrace on the EPC$START_EVENT routine call identifies the start and end for a particular instance of a duration event.

event ID

A numeric representation of a predefined event. Valid event identifiers range from 1 to 128.

event name

A 0- to 16-character string that names an event.

event pair

The matching start and end events which indicate the occurrence of a duration event.

event record

A record buffer used to pass the facility-specific items for an event to the DECtrace service routines.

facility

Software, usually referred to as a layered product, that serves a particular purpose and operates under VMS. See also **application program**

facility definition

A description of the events and items that DECtrace can capture for a particular facility. Each facility for which DECtrace can collect data must have a facility definition stored in the DECtrace administration database.

facility library

A text library which is referenced during DECtrace reinstallations. Facilities installed on your system before the installation of DECtrace can store their facility definitions in the facility library. The file specification for the library is SYS$COMMON:[SYSLIB]EPC$FACILITY.TLB.

facility selection

A description of what to collect during data collection. Facility selections include a list of facilities and their collection classes. One or more data collections can use the same facility selection.

file list

A file containing a list of file specifications to use as data capture files. Each file specification must be on a separate line within the file list. The default file type for the file list is TXT.

formatted data file

A file that contains data organized for reporting. The formatted data file can contain data from one or more data files.

history database

A single Rdb/VMS database that contains all of the informational and error messages associated with data collection. There is one history database per system or VAXcluster. You reference the history database with the logical name EPC$HISTORY_DB.

instrumenting

The act of adding DECtrace service routine calls to an application program so that event data can be collected.

interval

A time period when data collection takes place.

item

A numeric or string value that can be collected for an occurrence of an event. Up to 128 items can be captured for each event.

item flag

A 128-bit element of the item flags list. Each bit corresponds to an item that can be captured for a particular event for a facility. Each event has a 128-bit item flag associated with it.

item flag list

A 128-element array of item flags for all events for the facility. Each facility has its own item flag list.

item ID

A numeric representation of an item. Valid item identifiers range from 1 to 128.

item name

A 0- to 16-character string representing the name of an item.

local collection

A collection that is active or pending either on a standalone system or on one node in a VAXcluster as opposed to data collection scheduled cluster-wide.

local node

The system you are currently logged in to.

non-registered facility

A facility whose facility ID number is not registered with Digital. The facility ID for a non-registered facility is in the range of 2048 to 4095. Non-registered facilities usually represent customer-created applications. See also **facility**

point event

See **event**

registered facility

A facility whose facility ID number is registered with Digital. The facility ID for a registered facility is in the range of 1 to 2047. Registered facilities usually represent DIGITAL layered products. See also **facility**

registrar process

A detached process that handles all communication between the applications instrumented with DECtrace routine calls and the DECtrace administration database.

registration identifier

A 0- to 255-character string that is useful in distinguishing separate images use the same facilities. You use the /REGISTRATION_ID qualifier to the SCHEDULE COLLECTION command to collect data from processes with a specific registration ID.

remote

Pertaining to or originating from another node in the network.

reporting

The process of creating reports based on a formatted file or database. DECtrace can create reports based on data stored in a VAX Rdb/VMS database.

resource utilization items

A set of standard items that DECtrace collects for all facilities. The items are referenced by the group name RESOURCE_ITEMS.

service routines

A set of predefined DECtrace routines whose calls are instrumented in the source code of an application program so that event data can be collected.

source program

A program that expresses an algorithm in a programming language such as FORTRAN, COBOL, or assembly language.

start event

The beginning of a duration event. See **event**

Index

@ (Execute Procedure) command, 7–4

A

administration database, 1–3
  backing up, 9–5
ALL collection class
  See collection class

B

BIO, 6–5
buffered I/O operations, 6–5

C

CANCEL COLLECTION command
  example, 3–13
  format, 7–5
CAPACITY_PLANNING collection class
  See collection class
CDD/Plus, 4–2
class
  see collection class
cluster operations
  canceling data collection, 3–13
  scheduling data collection, 3–9
  starting DECtrace, 9–2
  stopping DECtrace, 9–3
collection class
  creating, 6–8
collection class (Cont.)
  definition, 2–2
  recommended types, 6–8
collection process, 1–4
Commands, 7–1tab
  CANCEL COLLECTION, 7–5
  CREATE DEFINITION, 7–8
  CREATE SELECTION, 7–21
  DELETE DEFINITION, 7–25
  DELETE SELECTION, 7–27
  @ (Execute Procedure), 7–4
  EXIT, 7–29
  EXTRACT DEFINITION, 7–30
  FORMAT, 7–32
  HELP, 7–39
  INSERT DEFINITION, 7–40
  REPORT, 7–42
  SCHEDULE COLLECTION, 7–54
  SET HISTORY, 7–59
  SHOW COLLECTION, 7–61
  SHOW DEFINITION, 7–63
  SHOW HISTORY, 7–65
  SHOW REGISTER, 7–68
  SHOW SELECTION, 7–69
  SHOW VERSION, 7–71
  SPAWN, 7–72
  STOP SYSTEM, 7–76
CPU usage, 6–5
CREATE DEFINITION command
  format, 7–8

CREATE DEFINITION command (Cont.)
- options
  - CLASS, 7–19
  - DEFAULT_CLASS, 7–20
  - EVENT, 7–16
  - GROUP, 7–15
  - ITEM, 7–11

CREATE SELECTION command
- example, 2–3
- format, 7–21

CURRENT_PRIO, 6–5

D

data collection file, 3–11
- file protection, 3–12

data merging, 4–2
deinstalling DECtrace, 6–13
DELETE DEFINITION command
- example, 6–9
- format, 7–25

DELETE SELECTION command
- example, 2–4
- format, 7–27

Detail Report, 4–5
DIO, 6–5
direct I/O operations, 6–5
disabling collection, 8–3, 9–6

E

EPC$DELETE_CONTEXT, 8–4
- example, 5–31

EPC$END_EVENT, 8–7
- example, 5–31

EPC$END_EVENTW, 8–11
EPC$EVENT, 8–12
EPC$EVENTW, 8–16
EPC$INIT, 8–17
- description of, 3–2
- example, 5–28
- instrumenting, 5–6

EPC$SET_CONTEXT, 8–23
- example, 5–30

EPC$START_EVENT, 8–26
- example, 5–31

EPC$START_EVENTW, 8–30
error handling, 8–2
error messages and recovery, 9–9
event-based collection, 1–1
events
- creating, 6–3

EXIT command, 7–29
EXTRACT DEFINITION command
- example, 6–10
- format, 7–30

F

facility definition
- creating, 6–2
- definition, 6–1
- deleting, 6–9
- displaying, 6–13
- options, 6–20

facility library, 6–13
facility selection
- See also CREATE SELECTION command
- creating, 2–1, 2–3
- deleting, 2–4
- displaying, 2–5

facility-specific items, 6–5
facility version, 6–2
file list, 3–12
file protection, 3–12
FORMAT command
- examples
  - Rdb/VMS database, 4–2
  - VAX RMS file, 4–3
- format, 7–32

Frequency Report, 4–9

G

guaranteeing data collection, 5–9

H

HELP command, 7–39
history database, 9–5

I

INSERT DEFINITION command
  example, 6–12
  format, 7–40
inserting definitions
  manually, 6–12
  using KITINSTAL.COM, 6–11
instrumenting an application, 5–1
  events and items, 5–13
  in VAX BLISS-32, 5–28
  in VAX C, 5–11
  in VAX COBOL, 5–8, 5–10
  in VAX FORTRAN, 1–6
  in VAX Pascal, 1–6
  linking, 5–32
  simple events, 5–12
items
  creating, 6–5

L

local collection, 3–9
local node, 3–9

M

merging data files, 4–2
multi-threaded facilities, 5–28

N

non-registered facility, 6–2

P

PAGEFAULTS, 6–5
PAGEFAULT_IO, 6–5
performance
  enhancing DECtrace, 9–1, 9–7
  instrumenting efficiently, 5–19
performance (Cont.)
  instrumenting for maximum efficiency, 5–20
PERFORMANCE collection class
  See collection class
priority of a process, 6–5

R

Rdb/VMS monitor process, 9–2, 9–3
registered facility, 6–2
Registrar process, 1–3, 3–2
  starting, 9–2
  stopping, 9–3
registration ID, 3–3
  See also SHOW REGISTER command
  facility-specific, 5–6
REPORT command
  format, 7–42
  options
    EVENT, 4–24, 7–47
    ITEM, 4–27, 7–51
    RESTRICTION, 4–28, 7–52
resource utilization items, 6–5
RESOURCE_ITEMS group, 6–5
RMU Monitor, 9–2, 9–3

S

sample application
  description, 1–5
  determining events, 5–5
  facility definition, 6–5
  instrumenting, 5–6
SCHEDULE COLLECTION command
  format, 7–54
scheduling data collection
  on a cluster, 3–9
  on a standalone system, 3–9
  on part of a cluster, 3–9
service routines, 8–1tab
  EPC$DELETE_CONTEXT, 8–4
  EPC$END_EVENT, 8–7
  EPC$END_EVENTW, 8–11
  EPC$EVENT, 8–12

service routines (Cont.)
  EPC$EVENTW, 8–16
  EPC$INIT, 8–17
  EPC$SET_CONTEXT, 8–23
  EPC$START_EVENT, 8–26
  EPC$START_EVENTW, 8–30
  return values, 8–2
SET HISTORY command, 7–59
SHOW COLLECTION command
  format, 7–61
SHOW DEFINITION command
  examples
    FULL format, 6–14
    NAMES_ONLY format, 6–19
  format, 7–63
SHOW HISTORY command
  example, 3–15
  format, 7–65
SHOW REGISTER command
  example, 3–5
  format, 7–68
SHOW SELECTION command
  examples
    BRIEF format, 2–5
    FULL format, 2–6
    NAMES_ONLY format, 2–7
  format, 7–69
SHOW VERSION command, 7–71
SPAWN command, 7–72
STOP SYSTEM command
  example, 9–3
  format, 7–76
Summary Report, 4–11

T

troubleshooting DECtrace problems, 9–7
  See also SHOW HISTORY command

V

virtual pages, 6–5
VIRTUAL_SIZE, 6–5

W

working set size, 6–5
WORKLOAD collection class
  See collection class
WS_GLOBAL, 6–5
WS_PRIVATE, 6–5
WS_SIZE, 6–5

How to Order Additional Documentation

Technical Support

If you need help deciding which documentation best meets your needs, call 800-343-4040 before placing your electronic, telephone, or direct mail order.

Electronic Orders

To place an order at the Electronic Store, dial 800-DEC-DEMO (800-332-3366) using a 1200- or 2400-baud modem. If you need assistance using the Electronic Store, call 800-DIGITAL (800-344-4825).

Telephone and Direct Mail Orders

Your Location	Call	Contact
Continental USA, Alaska, or Hawaii	800-DIGITAL	Digital Equipment Corporation P.O. Box CS2008 Nashua, New Hampshire 03061
Puerto Rico	809-754-7575	Local Digital subsidiary
Canada	800-267-6215	Digital Equipment of Canada Attn: DECdirect Operations KAO2/2 P.O. Box 13000 100 Herzberg Road Kanata, Ontario, Canada K2K 2A6
International	————	Local Digital subsidiary or approved distributor
Internal[1]	————	USASSB Order Processing - WMO/E15 *or* U.S. Area Software Supply Business Digital Equipment Corporation Westminster, Massachusetts 01473

[1]For internal orders, you must submit an Internal Software Order Form (EN-01740-07).

Reader's Comments

DECtrace for VMS
User's Guide
AA–PB90A–TE

Your comments and suggestions help us improve the quality of our publications.

Please rate the manual in the following categories:	Excellent	Good	Fair	Poor
Accuracy (product works as described)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Table of contents (ability to find topic)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page design (overall appearance)	☐	☐	☐	☐
Print quality	☐	☐	☐	☐

What I like best about this manual: ______________________________

____

What I like least about this manual: ______________________________

____

Additional comments or suggestions: ______________________________

____

____

I found the following errors in this manual:

Page	Description
______	______________________________
______	______________________________
______	______________________________

For which tasks did you use this manual?

☐ Installation	☐ Programming
☐ Maintenance	☐ System Management
☐ Marketing	☐ Training
☐ Operation/Use	☐ Other (please specify) ______________

Name/Title ______________________________

Company ______________________________

Address ______________________________

____

Phone ______________________ Date ______________

Fold Here and Tape

digital™

Please
Affix Stamp
Here

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications
55 Northeastern Blvd.
NUO1–1/G10
Nashua, NH 03062–3191

Fold Here

Cut Along Dotted Line

Reader's Comments

DECtrace for VMS
User's Guide
AA–PB90A–TE

Your comments and suggestions help us improve the quality of our publications.

Please rate the manual in the following categories:	Excellent	Good	Fair	Poor
Accuracy (product works as described)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Table of contents (ability to find topic)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page design (overall appearance)	☐	☐	☐	☐
Print quality	☐	☐	☐	☐

What I like best about this manual: ____________________

____

What I like least about this manual: ____________________

____

Additional comments or suggestions: ____________________

____

____

I found the following errors in this manual:

Page	Description
______	____________________
______	____________________
______	____________________

For which tasks did you use this manual?

☐ Installation	☐ Programming
☐ Maintenance	☐ System Management
☐ Marketing	☐ Training
☐ Operation/Use	☐ Other (please specify) ____________

Name/Title ____________________

Company ____________________

Address ____________________

____

Phone ____________________ Date ____________

Fold Here and Tape

digital™

Please
Affix Stamp
Here

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications
55 Northeastern Blvd.
NUO1–1/G10
Nashua, NH 03062–3191

Fold Here

Cut Along Dotted Line