VDE
VDE
Reference Manual

Order Number: ??–?????–??

December 1997

This is a reference for the OpenVMS Development Environment (VDE) source code control and source code builder package. This reference provides details of the OpenVMS Development Environment and the OpenVMS Source Control utilities; these utilities are used within various Digital engineering organizations including the OpenVMS group.

Revision/Update Information:	This manual supersedes the VDE Reference Manual, Version 1.5-0.
Software Version:	VDE Version 1.5-0
Operating System Version:	OpenVMS Alpha Version 6.2 OpenVMS VAX Version 6.2

Digital Equipment Corporation
Maynard, Massachusetts

11-DEC-1998 17:44:13.78

Copyright ©11-DEC-1998 17:44:13.78

The following are trademarks of Digital Equipment Corporation: Bookreader, DEC Notes, DECdirect, Digital, OpenVMS, OpenVMS Cluster, VAX, VAX Document, VMScluster, VMS, and the DIGITAL logo.

The following are third-party trademarks:

Motif, OSF, OSF/1, OSF/Motif, and Open Software Foundation are registered trademarks of the Open Software Foundation, Inc.

All other trademarks and registered trademarks are the property of their respective holders.

Contents

PREFACE ix

CHAPTER 1 INTRODUCTION TO VDE 1–1

1.1 USING VDE 1–1
1.1.1 Entering VDE and VSC Commands 1–1
1.1.1.1 Invoking VDE • 1–2
1.1.1.2 Entering Commands from DCL • 1–2
1.1.1.3 Using Abbreviations • 1–3
1.1.2 Using HELP 1–3

1.2 COMMAND SUMMARY 1–4

CHAPTER 2 VDE COMMANDS INVOKED FROM DCL 2–1
VDE DCL–2
VDE/DEPEND DCL–6
VDE/READ_UAF DCL–9
VDE/WHEREIS DCL–11
VSC DCL–14

CHAPTER 3 VDE COMMAND REFERENCE 3–1
@ VDE–2
ACCEPT REPLACEMENT VDE–3
ATTACH VDE–6
BUILD VDE–8
CANCEL DIRECTORY VDE–15
CANCEL FOLD VDE–18
CANCEL LOG VDE–20
CANCEL PRIVILEGES VDE–21
CANCEL REPLACEMENT VDE–22
CANCEL SESSION VDE–24
CANCEL VERIFY VDE–26
COMPILE VDE–27
CONVERT LIBRARY VDE–29
COPY VDE–31
CREATE ARCHITECTURE VDE–33
CREATE CONTEXT VDE–35
CREATE DIRECTORY_TREE VDE–38
CREATE FACILITY VDE–44
CREATE FOLD VDE–48

Contents

CREATE GROUP	VDE–51
CREATE LIBRARY	VDE–54
CREATE MODULE	VDE–65
CREATE REPLACEMENT	VDE–73
CREATE SCRIPT	VDE–75
CREATE SESSION	VDE–82
CREATE STEP	VDE–84
CREATE STREAM	VDE–89
CREATE USER	VDE–97
DEFINE/COMMAND	VDE–100
DEFINE/KEY	VDE–102
DELETE/COMMAND	VDE–106
DELETE/KEY	VDE–107
DELETE ARCHITECTURE	VDE–109
DELETE BUILD_JOB	VDE–110
DELETE CONTEXT	VDE–112
DELETE FACILITY	VDE–114
DELETE FOLD	VDE–116
DELETE GENERATION	VDE–118
DELETE GROUP	VDE–120
DELETE MODULE	VDE–123
DELETE REPLACEMENT	VDE–125
DELETE SCRIPT	VDE–127
DELETE STEP	VDE–130
DELETE STREAM	VDE–132
DELETE USER	VDE–134
DIFFERENCES	VDE–137
EXIT	VDE–142
EXTRACT SCRIPT	VDE–143
FETCH	VDE–146
HELP	VDE–151
INSERT GENERATION	VDE–152
INVOKE SCRIPT	VDE–155
LINK	VDE–159
MODIFY ARCHITECTURE	VDE–161
MODIFY BUILD_JOB	VDE–163
MODIFY CONTEXT	VDE–166
MODIFY FACILITY	VDE–169
MODIFY FOLD	VDE–171
MODIFY GENERATION	VDE–173
MODIFY GROUP	VDE–177
MODIFY LIBRARY	VDE–181
MODIFY MODULE	VDE–187
MODIFY REPLACEMENT	VDE–192
MODIFY RESERVATION	VDE–195
MODIFY SCRIPT	VDE–200
MODIFY SESSION	VDE–203
MODIFY STEP	VDE–205
MODIFY STREAM	VDE–209
MODIFY USER	VDE–215
PERFORM REPLACEMENT	VDE–218
POPULATE STREAM	VDE–221
REJECT REPLACEMENT	VDE–224

REMOVE GENERATION	VDE–227
REPLACE	VDE–229
RESERVE	VDE–239
REVIEW BUILD_JOB	VDE–247
REVIEW BUILD JOB COMMANDS	VDE–250
REVIEW REPLACEMENT	VDE–252
REVIEW REPLACEMENT COMMANDS	VDE–255
SCRIPT KEYWORDS	VDE–259
SCRIPT FUNCTIONS	VDE–267
SCRIPT TYPES	VDE–271
SET ARCHITECTURE	VDE–274
SET CONTEXT	VDE–275
SET DEVELOPER	VDE–277
SET DIRECTORY	VDE–278
SET EDITOR	VDE–284
SET FACILITY	VDE–287
SET KEY	VDE–288
SET LIBRARY	VDE–289
SET LOG	VDE–291
SET NOVERIFY	VDE–292
SET PRIVILEGES	VDE–293
SET PROMPT	VDE–296
SET STREAM	VDE–297
SET TERMINAL	VDE–298
SET VERIFY	VDE–299
SHOW ARCHITECTURE	VDE–300
SHOW BUILD_JOB	VDE–301
SHOW CONTEXT	VDE–306
SHOW DEFAULTS	VDE–308
SHOW DEVELOPER	VDE–309
SHOW DIRECTORY	VDE–310
SHOW EDITOR	VDE–314
SHOW FACILITY	VDE–316
SHOW FOLD	VDE–318
SHOW GENERATION	VDE–321
SHOW GROUP	VDE–326
SHOW KEY	VDE–329
SHOW LIBRARY	VDE–331
SHOW LOCATION	VDE–332
SHOW LOG	VDE–334
SHOW MEMORY	VDE–335
SHOW MODULE	VDE–337
SHOW PRIVILEGES	VDE–340
SHOW REPLACEMENT	VDE–342
SHOW RESERVATION	VDE–346
SHOW SCRIPT	VDE–349
SHOW SESSION	VDE–354
SHOW STEP	VDE–357
SHOW STREAM	VDE–360
SHOW SYMBOL	VDE–364
SHOW TERMINAL	VDE–365
SHOW USER	VDE–366
SHOW VERIFY	VDE–368

SHOW VERSION VDE–369
SPAWN VDE–370
START BUILD_JOB VDE–373
STOP BUILD_JOB VDE–376
STOP SUBPROCESS VDE–378
SUSPEND BUILD_JOB VDE–379
UNRESERVE VDE–381
VERIFY GENERATION VDE–384
WAIT BUILD_JOB VDE–388

APPENDIX A ASSOCIATED LOGICAL NAMES A–1

APPENDIX B RECORDING BUILD DEPENDENCIES B–1
LIB$REC_DEPENDENCY B–2

APPENDIX C CALLABLE INTERFACE C–1
VDE$FETCHMODULE C–3
VDE$REPLACEMODULE C–6
VDE$SETFACILITY C–8
VDE$SETLIBRARY C–11
VDE$SETSTREAM C–13
VDE$SHOWFACILITIES C–15
VDE$SHOWPROJECTS C–17
VDE$SHOWLIBRARY C–20
VDE$SHOWMODULES C–22
VDE$SHOWSTREAMS C–25
VDE$UNRESERVEMODULE C–28

APPENDIX D CMS LIBRARY UPLOADER D–1

APPENDIX E ERROR MESSAGES E–1

INDEX

TABLES

3–1	Key Names	VDE–102
3–2	Reasons	VDE–235
3–3	Script Types	VDE–271
3–4	Parameters	VDE–277
3–5	Privileges	VDE–293
A–1	Logical Names	A–1
B–1	Dependency Codes	B–2
C–1	Itemlist Codes	C–1
D–1	VDE$UPLOAD Syntax	D–1

Preface

Intended Audience

This manual is intended for users who want to know how to use the OpenVMS Development Environment.

Document Structure

This document contains a reference description of the OpenVMS Development Environment (VDE) source code control and source code build environment. It provides users with reference information related to module reservations and replacements, replacement reviews, library creation and various library maintenance operations.

This manual contains the following information:

- Chapter 1 provides a full description of the VDE Utility.
- Chapter 2 outlines the following VDE information:

  Invoking the utility
  Exiting from the utility
  Directing output
  Restrictions and VDE privileges required
- Chapter 3 contains the command reference for VDE commands. This section includes the command format, command qualifiers and parameters, and examples.
- Appendix B contains information useful for working directly with the VDE dependency recording callable interface.
- Appendix C contains information useful for working with the *unsupported* VDE calling interface.
- Appendix E includes a list of error messages that can be displayed by VDE.

Associated Documents

- Guide to Using VDE
- VDE Installation and New Features Guide

Conventions

In this manual, every use of OpenVMS Alpha means the OpenVMS Alpha operating system, every use of OpenVMS VAX means the OpenVMS VAX operating system, and every use of OpenVMS means both the OpenVMS Alpha operating system and the OpenVMS VAX operating system.

The following conventions are used in this manual:

Ctrl/*x*	A sequence such as Ctrl/*x* indicates that you must hold down the key labeled Ctrl while you press another key or a pointing device button.
PF1 *x*	A sequence such as PF1 *x* indicates that you must first press and release the key labeled PF1, then press and release another key or a pointing device button.
GOLD *x*	A sequence such as GOLD *x* indicates that you must first press and release the key defined GOLD, then press and release another key. GOLD key sequences can also have a slash (/), dash (-), or underscore (_) as a delimiter in EVE commands.
Return	In examples, a key name enclosed in a box indicates that you press a key on the keyboard. (In text, a key name is not enclosed in a box.)
. . .	A horizontal ellipsis in examples indicates one of the following possibilities: • Additional optional arguments in a statement have been omitted. • The preceding item or items can be repeated one or more times. • Additional parameters, values, or other information can be entered.
. . .	A vertical ellipsis indicates the omission of items from a code example or command format; the items are omitted because they are not important to the topic being discussed.
( )	In format descriptions, parentheses indicate that, if you choose more than one option, you must enclose the choices in parentheses.
[ ]	In format descriptions, brackets indicate optional elements. You can choose one, none, or all of the options. (Brackets are not optional, however, in the syntax of a directory name in an OpenVMS file specification, or in the syntax of a substring specification in an assignment statement.)
{ }	In format descriptions, braces surround a required choice of options; you must choose one of the options listed.
boldface text	Boldface text represents the introduction of a new term or the name of an argument, an attribute, or a reason. Boldface text is also used to show user input in Bookreader versions of the manual.
italic text	Italic text emphasizes important information, indicates variables, and indicates complete titles of manuals. Italic text also represents information that can vary in system messages (for example, Internal error *number*), command lines (for example, /PRODUCER=*name*), and command parameters in text.
UPPERCASE TEXT	Uppercase text indicates a command, the name of a routine, the name of a file, or the abbreviation for a system privilege.

-	A hyphen in code examples indicates that additional arguments to the request are provided on the line that follows.
numbers	All numbers in text are assumed to be decimal, unless otherwise noted. Nondecimal radixes—binary, octal, or hexadecimal—are explicitly indicated.
mouse	The term *mouse* refers to any pointing device, such as a mouse, a puck, or a stylus.
MB1, MB2, MB3	MB1 indicates the left mouse button, MB2 indicates the middle mouse button, and MB3 indicates the right mouse button. (The buttons can be redefined by the user.)
PB1, PB2, PB3, PB4	PB1, PB2, PB3, and PB4 indicate buttons on the puck.
SB, SB	SB and SB indicate buttons on the stylus.

1 Introduction to VDE

The OpenVMS Development Environment (VDE) utilities assist OpenVMS engineers in managing the software development process.

This is the VDE reference manual. For a complete introduction, overview of the VDE package, and for information on commonly performed VDE tasks, please see the *Guide to Using VDE*.

VDE commands enable OpenVMS engineers to perform the following tasks:

- Create, modify, display, and delete information in a VDE database such as that used by OpenVMS. This information includes the users, facilities, modules, development streams, groups, and other entities that define the software system that you want to manage and build.
- Define, set, and show current context and default settings.
- Build a software system. VDE can perform both complete and minimal system builds. In a complete build, all modules in the software system are compiled, linked, and otherwise processed. In a minimal system build, only those modules that have been changed since the last system build are compiled, linked, and processed.
- Reserve source modules from the VDE library and replace them after they have been modified.
- Collect and report software development statistics.
- Transfer control to another process while still in VDE. Users typically interact with the VDE package using either the VDE or the VSC DCL commands. Both commands have equivilent capabilities, though the VSC defaults are specifically tailored to the requirements of OpenVMS engineering.

VDE is implemented as a set of images, and several DCL commands.

1.1 Using VDE

This section describes how to enter commands from within the utility or from the DCL command line. It also describes how to use the VDE help facility.

1.1.1 Entering VDE and VSC Commands

You can enter VDE commands in either of two ways: by invoking the utility and allowing it to prompt for input, or by including the command directly on the DCL command line.

There are two VDE interfaces supported within the VDE package: the VDE interface, and the VSC interface. Both interfaces support the same command verbs, and both function similarly.

The only salient differences between the two interfaces involves differences in the invocation defaults and the prompt string: VDE is a general interface and does not default to a particular source code library, while VSC is specifically tailored for users accessing the OpenVMS source code libraries and sets its library default appropriately.

Save for the default options and cosmetic aspects such as the command prompt string, the VDE and VSC interfaces are fully compatible and are fully interchangeable.

Note that certain qualifiers present on the VDE utility—such as VDE/READ_UAF and VDE/STATUS—may invoke utilities other than the VDE utility. There may be no VSC equivilents to some of these VDE utilities.

1.1.1.1 Invoking VDE

To invoke the VDE utility, enter the VDE or the VSC command at the OpenVMS DCL prompt ($). OpenVMS responds with the appropriate prompt for the utility.

```
$ VDE
VDE> EXIT
$ VSC
VSC> EXIT
$
```

Once the command has been invoked, you then can enter commands at the VDE prompt. For example, to display the default VDE library, enter the SHOW LIBRARY command:

```
VDE> SHOW LIBRARY
Main library LIBNAME in root directory DEV$:[PROJDIR]
VDE>
```

You are returned to the VDE prompt when the command has executed and you then can enter another VDE command.

Ending a Session

To exit from VDE, enter the VDE command EXIT or press CTRL/Z.

1.1.1.2 Entering Commands from DCL

To enter a VDE command from the DCL command line, type VDE at the DCL prompt ($) then type a VDE command on the same DCL command line. The following example shows how to display the name of the default VDE library:

```
$ VDE SHOW LIBRARY
Main library LIBNAME in root directory DEV$:[PROJDIR]
$
```

You are returned to the DCL prompt ($) when the command has been executed. For this reason, this form is most appropriate when you only need to enter a single VDE command.

Multiple commands can be specified on a single VDE or VSC DCL command line by using a semicolon as a delimiter.

```
VDE> SHOW LIBRARY; SHOW VERSION
Library VMS in root directory VSC$ROOTDISK:[VMS.DBROOT]
Executing VDE for OpenVMS Alpha Version V1.5-0
VDE>
```

1.1.1.3 Using Abbreviations

All command and qualifier keywords for VDE commands can be abbreviated to the shortest unique abbreviation. The DELETE command, for example, can be abbreviated to DEL because no other VDE command begins with those three letters. The DELETE command cannot be abbreviated to DE, however, because the DEFINE command makes the two-letter abbreviation ambiguous.

Note: The number of letters needed to reference a particular VDE command is subject to change in future releases of VDE. DIGITAL strongly recommends that all VDE or VSC commands imbedded within command procedures specify the full command. This avoids errors should any particular VDE command abbreviation become ambiguous in a potential future release of the VDE package.

1.1.2 Using HELP

The VDE interfaces contain a HELP command and an extensive online HELP facility. To obtain help on a particular command, append the command to the HELP command.

```
VDE> HELP SHOW VERSION

SHOW

  VERSION

       Displays the VDE version string for the version of VDE you are
       running.

       Format

         SHOW VERSION


    Additional information available:

    Parameters Qualifiers Examples

SHOW VERSION Subtopic?
```

The help library also contains a complete list of all error messages and the associated recovery procedures.

1.2 Command Summary

The following table lists all the available VDE commands by functional category:

Creating Database Information	Modifying Database Information
CREATE CONTEXT	MODIFY CONTEXT
CREATE DEPENDENCY	MODIFY DEPENDENCY
CREATE FACILITY	MODIFY FACILITY
CREATE GROUP	MODIFY GROUP
CREATE LIBRARY	MODIFY LIBRARY
CREATE MODULE	MODIFY MODULE
CREATE SESSION	MODIFY SESSION
CREATE STREAM	MODIFY STREAM
CREATE USER	MODIFY USER

Displaying Database Information	Deleting Database Information
SHOW CONTEXT	DELETE CONTEXT
SHOW DEPENDENCY	DELETE DEPENDENCY
SHOW FACILITY	DELETE FACILITY
SHOW GROUP	DELETE GROUP
SHOW LIBRARY	DELETE LIBRARY
SHOW MODULE	DELETE MODULE
SHOW SESSION	UNRESERVE/SESSION
SHOW STREAM	DELETE STREAM
SHOW USER	DELETE USER

Setting the Development Context	
SHOW DEFAULTS	SET STREAM
SET LIBRARY	SET PRIVILEGES
SHOW LIBRARY	CANCEL PRIVILEGES
SET CONTEXT	SHOW PRIVILEGES
SET FACILITY	RESERVE/SESSION
CREATE CONTEXT	MODIFY CONTEXT
SHOW CONTEXT	DELETE CONTEXT
SHOW SESSION	UNRESERVE/SESSION

Reserving Modules	Building the System
RESERVE	BUILD
REPLACE	COMPILE
UNRESERVE	LINK
FETCH	TEST

Defining Directory Structures	
SET DIRECTORY	CANCEL DIRECTORY
SHOW DIRECTORY	CREATE DIRECTORY_TREE

Mode Settings	
SHOW ALL	SET VERIFY
SET LOG	SET NOVERIFY
CANCEL LOG	CANCEL VERIFY
SHOW LOG	SHOW VERIFY
SET TERMINAL	
SHOW TERMINAL	

Tailoring VDE	Exiting or Transferring Control
DEFINE/COMMAND	HELP
DELETE/COMMAND	*@file-name*
SHOW SYMBOL	SPAWN
DEFINE/KEY	ATTACH
DELETE/KEY	EXIT
SET KEY	
SHOW KEY	
SET PROMPT	

2 VDE Commands Invoked from DCL

VDE

The OpenVMS Development Environment (VDE) Utility is a tool that software developers can use to manage software development. VDE uses a relational database to store development information about your software system and to manage multiple instances of that system. VDE also manages the disk directories, CMS libraries, and files that comprise your software system. In managing a software system, VDE can perform the following functions:

- Store and retrieve information about the modules, facilities (groups of modules), module generations, development streams, and other entities that make up your software system.
- Reserve source modules from the VDE library and replace them after they have been modified. Modules can be reserved from and replaced into multiple development streams and changes can be automatically propagated between streams.
- Build the software system. VDE can perform both full and minimal system builds. In a full build, all modules in the software system are compiled, linked, or otherwise processed. In a minimal system build, only those modules that have been changed since the last system build are compiled, linked, or processed.
- Collect and report software development statistics.

VDE includes extensive built-in HELP. At the VDE>, type "HELP". An extensive documentation set, in Bookreader and Postscript formats, is also available.

FORMAT

VDE *[vde-command [; vde-command...]]*

PARAMETER

vde-command

A VDE command to be performed. The available VDE commands are described in Chapter 3. You can specify multiple VDE commands if you separate them by semicolons (;).

This parameter is optional; it you omit it, VDE prompts you for commands.

DESCRIPTION

The VDE command invokes the VDE utility and allows you to enter VDE subcommands. Specifying the VDE command with specific qualifiers enables you to modify the VDE environment or invoke separate programs that are part of the VDE system.

To invoke the VDE utility, enter the VDE command at the DCL prompt ($). The system responds with the prompt for the VDE utility (VDE>). For example:

```
$ VDE
VDE>
```

At the VDE> prompt, you can enter VDE subcommands to perform VDE functions. For simplicity, these subcommands are referred to as ''VDE commands'' in the rest of this description. To exit from VDE, enter the EXIT command or press CTRL/Z.

The following example shows how to enter VDE commands from within the utility:

```
$ VDE
VDE> SHOW LIBRARY
Main library LIBNAME in root directory DEV$:[PROJDIR]
VDE> EXIT
$
```

You can also enter VDE commands directly on the VDE command line. In this case, VDE does not prompt you for commands. The following example shows how to enter a VDE command from the DCL command line:

```
$ VDE SHOW LIBRARY
Main library LIBNAME in root directory DEV$:[PROJDIR]
$
```

You can also enter the following commands from the DCL command line (either interactively or from within a coded file called a ''VDE script'') to invoke separate programs that are part of the VDE system:

- VDE/DEPEND
- VDE/READ_UAF
- VDE/WHEREIS

These commands are described separately in this chapter.

To use VDE, you must specify the VDE library that you want to use. The VDE library for a software system consists of a root directory, a VDE database residing in that root directory, and a directory tree that contains all CMS libraries, source files, and other files that make up your software system. You specify the VDE library you want to use by giving the directory specification for the library root directory. The root directory contains the database and the database contains pointers to all other directories that make up the library.

Unless you are creating a new VDE library, you can specify the library using any of three methods. First, you can define a logical name VDE$LIBRARY to give the library's root directory. If you use the same VDE library most of the time, you should define VDE$LIBRARY in your login file. Second, you can use the /LIBRARY qualifier to the VDE command to specify the library mnemonic or root directory. And third, you can use the VDE SET LIBRARY command to specify the mnemonic or root directory. You must use one of these three methods to specify the VDE library.

QUALIFIERS

/INTERFACE=keyword

The /INTERFACE qualifier allows the user to select a keyword representing the prefered user interface presented by VDE. With the VDE/INTERFACE=DECWINDOWS keyword, the Motif graphical user

interface is used. With the VDE/INTERFACE=CHARACTER_CELL keyword, the traditional command interface is used.

/KEPT_PROCESS
/NOKEPT_PROCESS (default)

Specifies whether VDE runs with or without creating a kept process. /KEPT_PROCESS specifies that the VDE utility should run in a kept subprocess that is retained from one invocation of the VDE command to the next. Use of a kept subprocess allows VDE to connect to the VDE database once for many invocations of VDE, thus speeding up subsequent accesses to the database. It also allows VDE to retain state information, such as the default stream and facility, between invocations. /NOKEPT_PROCESS specifies that VDE not run in a kept subprocess.

Do *not* attempt to use the DCL ATTACH command to attach to a VDE or VSC kept process.

/LIBRARY=directory-spec

Specifies the mnemonic name or root directory of the VDE library with which you want to work. Subsequent VDE operations will access the database associated with that library.

/DEPEND lang ana-file [in-out-file [,in-out-file...]]

The /DEPEND qualifier causes VDE to invoke a separate program that automatically collects build dependency information. The VDE/DEPEND combination is described as a separate command later in this chapter.

/READ_UAF

The /READ_UAF qualifier causes VDE to invoke a separate program used to update the User Table in a VDE database from information in the system user authorization file (UAF). The VDE/READ_UAF combination is described as a separate command later in this chapter.

EXAMPLES

1

```
$ DEFINE VDE$LIBRARY DEV$:[PROJDIR]
$ VDE
VDE> SHOW LIBRARY
Main library LIBNAME in root directory DEV$:[PROJDIR]
VDE> SHOW STREAM V2.1-1
Stream V2.1-1          "First update to version 2.1"
VDE> HELP
...help text displayed...
VDE> EXIT
$
```

This example defines logical name VDE$LIBRARY to specify the root directory of the VDE library you want to work with. This directory contains the VDE database for the library. The VDE command entered from DCL then gives the VDE> prompt. You can now enter VDE commands, in this case the SHOW LIBRARY and SHOW STREAM commands. The EXIT command ends the VDE session and returns you to the DCL prompt ($).

2
```
$ VDE SHOW LIBRARY; SHOW STREAM V2.1-1
Main library LIBNAME in root directory DEV$:[PROJDIR]
Stream V2.1-1          "First update to version 2.1"
$
```

This example shows two VDE commands entered directly on the DCL-level VDE command line. Both commands are executed, after which VDE exits and returns you to the DCL prompt ($). In this case, VDE does not prompt for commands. The example assumes that logical name VDE$LIBRARY has already been defined, perhaps in the user's login file.

3
```
$ VDE/LIBRARY=DEV$:[ROOTDIR]
VDE> SHOW LIBRARY
Library LIBNAME in root directory DEV$:[ROOTDIR]
VDE> EXIT
$
```

In this example, the /LIBRARY qualifier specifies that the root directory for the VDE library is DEV$:[ROOTDIR]. Entering the SHOW LIBRARY command at the VDE prompt confirms that this library has been set.

4
```
$ VDE
VDE> SET LIBRARY DEV$:[ROOTDIR]
VDE> SHOW LIBRARY
Library LIBNAME in root directory DEV$:[ROOTDIR]
VDE> EXIT
$
```

This example uses the SET LIBRARY command to specify the root directory of the VDE library to use. The SHOW LIBRARY command confirms that this library has been set.

VDE/DEPEND

Invokes a program that automatically collects build dependency information from a compiler-generated analysis file to determine the inputs and outputs of a build step.

This command should only be invoked in DCL command procedures created from VDE scripts to perform build steps. Do not attempt to run this program interactively.

FORMAT

VDE/DEPEND *lang ana-file [in-out-file [,in-out-file...]]*

PARAMETERS

lang

The name of the language processor that generated the Source Code Analyzer analysis file. This is a required parameter. Keywords for the supported language processors are as follows:

Keyword	Language Processor
ADA	Ada compiler
B32	Bliss compiler
C	C compiler
DML	DBMS preprocessors for compilers
FOR	Fortran compiler
MAR	Macro assembler
NOANA	No analysis file given (only the **in-out-file** parameters are used)
PLI	PLI compiler
PAS	Pascal compiler
RBA	RDML preprocessor for BASIC
RBL	RDML preprocessor for Bliss
RC	RDML preprocessor for C
RCO	RDML preprocessor for COBOL
RFO	RDML preprocessor for FORTRAN
RPA	RDML preprocessor for Pascal
REQ	Bliss library compilation
SC	SQL preprocessor for C
SCO	SQL preprocessor for COBOL
SFO	SQL preprocessor for FORTRAN
SPA	SQL preprocessor for Pascal
SPL	SQL preprocessor for PL/I
SQLADA	SQL preprocessor for Ada

Keyword	Language Processor
SQLMOD	SQL module language

ana-file

The file specification for the analysis file containing the dependency information. This is a required parameter. If the **lang** parameter is NOANA, specify two double quotes ("") to indicate that the **ana-file** parameter is null.

in-out-file

A file specification that specifies an input file or output file for the build step. The file specification can be followed by the /INPUT qualifier to indicate that the file is an input to the step or by the /OUTPUT qualifier to indicate that the file is an output of the step. If neither qualifier is specified, the file is assumed to be an input file. This parameter is used to record input and output files that are not specified in the analysis file or that are not correctly classified as inputs or outputs from the information in the analysis file.

DESCRIPTION

The VDE/DEPEND command invokes a separate program that automatically collects build dependency information from a compiler-generated Source Code Analyzer (SCA) analysis file. The VDE/DEPEND command is used in command procedures generated from VDE scripts to compile source modules. The program reads the analysis (.ANA) file along with information specified on the command line to determine all input and output modules for the current build step. This information is written to the VDE database if the build step completes successfully.

The analysis file specifies all input and output files of the compilation but does not indicate which files are inputs and which files are outputs. VDE/DEPEND therefore requires that you specify the language that was compiled so that the program can guess which files are inputs and which are outputs based on file types. As a result, it is important that you use the standard file types for all of your compilations.

Because the analysis file and command line supply VDE/DEPEND with a list of file names, VDE must translate those file names into VDE module names before it can record build dependencies in its database. To do so, VDE uses the following rules:

- VDE assumes that the VDE module name and type are the same as the file name and type from the analysis file or command line.
- VDE looks up the full directory specification of the file in the VDE database. If that directory is a subdirectory in the VDE directory tree for the stream being built, then VDE can determine the facility of the module. The facility is determined from the part of the directory tree in which the file is found.
- Once the facility, module name, and module type are determined, the module can be looked up in the VDE database. If the module is an output module for the build step and the module is not found in the database, VDE adds it to the database as a new derived module.

- If the full directory specification cannot be found in the VDE database, the module is assumed to be outside of your VDE library and is ignored. As a result, input modules from system libraries, for example, are omitted from VDE's dependency graph.

As a result of these rules, it is important that each source module or derived module be stored somewhere in the directory tree for the stream you are building and, within that stream, for the facility to which the module belongs. This is a constraint that your scripts must obey; within that constraint, scripts are free to place files anywhere you want.

EXAMPLES

1

```
$ DEFINE SRC$ DEV$:[ROOTLIB.MAIN.FAC.SRC]
$ DEFINE OBJ$ DEV$:[ROOTLIB.MAIN.FAC.OBJ]
$ VDE FETCH [FAC]FOO.C/OUTPUT=SRC$:
$ CC/ANA=OBJ$:/OBJ=OBJ$:  SRC$:FOO.C
$ VDE/DEPEND C OBJ$:FOO.ANA
```

This example shows the lines generated from a VDE script to compile module FOO.C. The first two lines define logical names SRC$ and OBJ$ to point to the source and object directories for stream MAIN and facility FAC within the VDE library. The third line fetches source module FOO.C from the VDE library into the source directory. The next line runs the C compiler, compiling module FOO.C to produce an object file FOO.OBJ and an SCA analysis file FOO.ANA. The last line runs VDE/DEPEND to read that analysis file. From information in the analysis file, VDE/DEPEND determines what input files were used and what output files were produced by the compilation. This information will be passed to VDE for inclusion in the VDE database.

2

```
$ VDE/DEPEND REQ OBJDIR:LIBDEF.ANA OBJDIR:LIBDEF.L32/OUTPUT
```

When the command procedure containing this line executes, VDE/DEPEND will read the analysis file OBJDIR:LIBDEF.ANA to determine the input and output modules of a Bliss library compilation. Because a Bliss library compilation produces one .L32 module as an output and may use many .L32 modules as inputs, the VDE/DEPEND command line must specify which .L32 module is the one output. This is done by specifying OBJDIR:LIBDEF.L32 with the /OUTPUT qualifier at the end of the command. This is an example of using an **in-out-file** parameter to override the default assumption that a module of that type is an input to the compilation.

3

```
$ VDE/DEPEND NOANA "" SRCDIR:FOO.MSG/INPUT,OBJDIR:FOO.EXE/OUTPUT
```

This example shows how VDE/DEPEND can be used to specify input and output modules when no analysis file is available. The message compiler does not produce an analysis file. For this reason, the command uses the NOANA keyword and specifies the null string as the analysis file parameter. The actual input and output files are specified by a list of **in-out-file** parameters at the end of the command.

VDE/READ_UAF

Updates the User Table in a VDE database from information in the system user authorization file (SYSUAF).

FORMAT VDE/READ_UAF[/WRITE_UAF]

DESCRIPTION

The VDE/READ_UAF command invokes a separate program that updates the User Table in a VDE database from information in the system user authorization file (SYSUAF). For each user recorded in the VDE database, the program looks up that user in the UAF to determine the user's full name from the UAF owner field. That name is then stored in the VDE database if not already there. The program also determines whether the user's account is still valid. If not, the user is marked as deleted in the VDE database so that the database is no longer accessible to that user.

The VDE/READ_UAF command also supports a /WRITE_UAF qualifier. Specification of VDE/READ_UAF/WRITE_UAF causes VDE to update the process quota values of all VDE users in the System Authorization Database (SYSUAF). Any process quotas below the minimum VDE requirements are increased, but quotas above the minimum are not altered. The /WRITE_UAF qualifier is valid only when specified with /READ_UAF, and it requires read and write access to the SYSUAF database.

VDE/READ_UAF can be run at regular intervals to maintain the VDE database so it stays synchronized with the system authorization database. This way, you do not have to manually maintain the VDE User Table with the CREATE USER, MODIFY USER, and DELETE USER commands. Specification of the the target VDE library is required via the /LIBRARY qualifier on the command, or via the logical name VDE$LIBRARY.

To use this command, you must have read access to the SYSUAF system authorization file. Access to the system authorization file can be granted via OpenVMS an privilege such as READALL or BYPASS, via an ACL on the SYSUAF file, or via various other means. Contact your system manager for assistance.

QUALIFIERS

/DELETE

The /DELETE qualifier causes VDE to delete any VDE usernames that are not also found in SYSUAF. This is not generally desirable, and can potentially lead to problems should SYSUAF be misdirected.

/WRITE_UAF

The /WRITE_UAF qualifier causes VDE to update any process quotas stored in the SYSUAF file that are below the required minimum settings to the minumum settings. Quota values above the minimum settings are not altered.

EXAMPLE

```
$ SET PROCESS/PRIVILEGE=BYPASS
$ VDE/READ_UAF/WRITE_UAF/LIBRARY=DEV$:[VDELIBDIR]
$ SET PROCESS/PRIVILEGE=NOBYPASS
```

The first command sets the OpenVMS privilege BYPASS, which overrides all protections on the system authorization database. The second command runs VDE/READ_UAF to update the VDE database from the user authorization file, and the system authorization database process quotas from the list of users in the VDE database. The final line removes the BYPASS privilege to clean up.

VDE/WHEREIS

Invokes a utility that retrieves site-specific configuration information from VDE.

No direct access nor direct retrieval of any underlying configuration information should be performed, nor should the underlying VDE$LIBRARY_* logical names be translated directly by command procedures, nor should software depend on the continued accessability of the underlying information.

FORMAT VDE/WHEREIS

DESCRIPTION

The VDE/WHEREIS command invokes a separate utility that retrieves site-specific configuration information associated with a VDE library. The command returns such information as the VDE mnemonic name and location of the Rdb database and CMS libraries associated with the VDE library. VDE/WHEREIS can be used from the command line or within DCL scripts to facilitate library management operations independent of specific configuration information.

VDE/WHEREIS returns configuration information via a set of VDE$WHEREIS_*info-item* local symbols or process logical names. Whether symbols or logical names are used is controlled by the /RETURN qualifier. The local symbols/process logicals currently defined by VDE/WHEREIS are described below:

VDE$WHEREIS Symbols/Logical Names

VDE$WHEREIS_MNEMONIC
: VDE shorthand name associated with a database

VDE$WHEREIS_BACKUPS_CURRENT
: Location of the latest Rdb database backups

VDE$WHEREIS_BACKUPS_ARCHIVE
: Location of the archival Rdb database backups

VDE$WHEREIS_BACKUPS_AFTER_IMAGE
: Location of the Rdb database after-image journals

VDE$WHEREIS_CMS
: Root specification of the CMS libraries associated with a database

VDE$WHEREIS_DATABASE
: Location (root) of the Rdb database file

VDE$WHEREIS_LABEL
: VDE text label (remark string) associated with a database

VDE$WHEREIS_BACKUP_INCR
: File name of current Rdb database incremental backup saveset

VDE/WHEREIS

VDE$WHEREIS Symbols/Logical Names
VDE$WHEREIS_BACKUP_FULL
File name of current Rdb database full backup saveset
VDE$WHEREIS_STATISTICS
Specification of the VDE statistics database file
VDE$WHEREIS_STAGING
Location (root) where VDE stages library changes
VDE$WHEREIS_BUILD
For future use/not fully implemented

When invoking VDE/WHEREIS, you must specify a VDE library, either via the /LIBRARY qualifier with a library mnemonic or root, or via the /INDEX qualifier with the library index.

QUALIFIERS

/INDEX=library-index
/LIBRARY=library-spec

The /INDEX and /LIBRARY qualifiers to VDE/WHEREIS are used to identify a VDE library for which the utility should return configuration information.

The /INDEX qualifier is used to identify a library by index, or by the offset into the list of all available VDE libraries present on the system. The first library in the list has an index of 0. This qualifier can be used to obtain configuration information for all VDE libraries on a given system, by starting with index 0 and incrementing the index until VDE/WHEREIS completes with a failure status.

The /LIBRARY qualifier to VDE/WHEREIS is used to identify a VDE library via either the shorthand mnemonic name or the database root directory, just as with the standard VDE command.

The /INDEX and /LIBRARY qualifiers are mutually exclusive. If both are specifed, the /INDEX qualifier is used. One of either /INDEX or /LIBRARY is *required* with the VDE/WHEREIS command.

/LOG

Controls whether informational messages are printed to display the requested library information as it is retrieved. By default, no messages are printed.

/RETURN=keyword

The /RETURN qualifier is used to specify the method in which site-specific library information should be returned to the caller. The qualifier accepts the following two *keyword* parameters:

- SYMBOLS - the requested information is returned via the local symbols VDE$WHEREIS_*info-item*. This is the default behavior.
- LOGICAL_NAMES - the requested information is returned via the process logical names VDE$WHEREIS_*info-item*.

If no /RETURN qualifier is specified with the VDE/WHEREIS command, /RETURN=SYMBOLS is assumed.

EXAMPLES

1

```
$ VDE/WHEREIS /LIBRARY=PROD$LIB /RETURN=LOGICAL_NAMES
$ DIRECTORY/DATE/SINCE=10-OCT-1996 VDE$WHEREIS_CMS:[*.CMS.CMS$000]

Directory VMS$:[PROD$SHR.CMS.CMS$000]

PROD$SHR.OPT;29      10-OCT-1996 16:45:01.74
PRODDEF.H;15         15-OCT-1996 15:12:01.21
PRODIO.C;24          14-OCT-1996 22:39:24.73
PRODUTIL.C;5         16-OCT-1996 16:31:00.79
   .
   .
   .
```

This example shows how VDE/WHEREIS can be used to retrieve information about a VDE library's CMS-level source files, without necessarily knowing the location of the CMS library.

2

```
$ vde_lib_index = F$EDIT(p1,"UPCASE,TRIM")
$ VDE/WHEREIS /INDEX='vde_lib_index' /RETURN=SYMBOLS
$ IF .NOT. $STATUS THEN GOTO BCKLIB$Abort
$ DEFINE/PROCESS BCKLIB$DB_SOURCE   'vde$whereis_database'
$ DEFINE/PROCESS BCKLIB$BACKUP_AREA 'vde$whereis_backups_current'
$ WRITE SYS$OUTPUT " Performing full database backup for:"
$ WRITE SYS$OUTPUT "    ''vde$whereis_mnemonic' library .."
$ RMU /BACKUP /LOG /ONLINE BCKLIB$DB_SOURCE:VDE$DATABASE.RDB -
     MLIB$BACKUP_AREA:'vde$whereis_backup_full'.RBF
```

This example shows how VDE/WHEREIS can be used within a DCL command procedure to perform a full backup of the Rdb database associated with a VDE library, independent of the specific library or the database location.

3

```
$ VDE/WHEREIS /LIBRARY=EVMS /LOG
VDE Ancillary Support for OpenVMS Alpha Version V1.5-0
%VDE-I-WHEREIS, component VDE$WHEREIS_MNEMONIC is EVMS
%VDE-I-WHEREIS, component VDE$WHEREIS_BACKUPS_CURRENT is DISK$VDERDBBKP:[VDE.MASTERPACK_BACKUPS.CURREI
%VDE-I-WHEREIS, component VDE$WHEREIS_BACKUPS_ARCHIVE is DISK$VDERDBBKP:[VDE.MASTERPACK_BACKUPS.OLD]
%VDE-I-WHEREIS, component VDE$WHEREIS_BACKUPS_AFTER_IMAGE is DISK$VDERDBBKP:[VDE.MASTERPACK_BACKUPS.DI
%VDE-I-WHEREIS, component VDE$WHEREIS_CMS is VMS$
%VDE-I-WHEREIS, component VDE$WHEREIS_DATABASE is DISK$VDEROOT:[VDE.VMS.DBROOT]
%VDE-I-WHEREIS, component VDE$WHEREIS_LABEL is OpenVMS Alpha Base Operating System
%VDE-I-WHEREIS, component VDE$WHEREIS_BACKUP_INCR is VDE$DATABASE$EVMS_BCKP_INCR
%VDE-I-WHEREIS, component VDE$WHEREIS_BACKUP_FULL is VDE$DATABASE$EVMS_BCKP_FULL
%VDE-I-WHEREIS, component VDE$WHEREIS_STATISTICS is DISK$VDERDBBKP:[VDE.CHECKIN_STATISTICS]STATISTICS_
%VDE-I-WHEREIS, component VDE$WHEREIS_STAGING is DISK$VDESTAGE:[VDE.VMS.VDE$STAGING]
```

This example shows how the /LOG qualifier to the VDE/WHEREIS command can be used to display current site-specific library information.

VSC

The VSC command is a command variant that by default provides source control services for the OpenVMS VAX library. Use the VSC command to perform the following operations:

- Reserve source modules from the OpenVMS VAX library for modification
- Replace source modules to the OpenVMS VAX library after they have been modified
- Review source modules modified by other developers
- Create new modules, facilities, and development streams in the OpenVMS VAX library
- Administer the OpenVMS VAX library and control its use
- Display information about the OpenVMS sources in the OpenVMS VAX library

Use the VSC HELP command for additional information on specific VSC commands.

The VSC command runs the OpenVMS Development Environment (VDE) utility and accepts the same commands as the VDE command, but it automatically accesses the VDE database for the OpenVMS VAX library. Unlike VSC, the VDE command does not automatically access the OpenVMS VAX library. (Further information on the VDE command is available.)

VSC includes extensive built-in HELP. At the VSC>, type "HELP". An extensive documentation set, in Bookreader and Postscript formats, is also available.

FORMAT

VSC *[VDE-command] [;VDE-command]*

DESCRIPTION

The VSC command performs source control operations for the OpenVMS VAX source code control library. It runs the same image as the VDE command and differs only in that it automatically operates on the OpenVMS VAX library, and not any of the other VDE libraries around. Therefore, at the VSC prompt (VSC>), you can enter any valid VDE command.

To invoke the VSC utility, enter the VSC command at the DCL prompt ($). The system responds with the prompt for the VSC utility (VSC>). For example:

```
$ VSC
VSC>
```

At the VSC> prompt, you can enter VSC commands to perform VSC functions. To exit from VSC, enter the EXIT command or press CTRL/Z. The following example shows how to enter VSC commands from within the utility at the VSC> prompt:

```
$ VSC
VSC> SHOW FACILITY SYS
Facility SYS             "SYS Facility"
VSC> EXIT
$
```

You can also enter VSC commands directly on the VSC command line. The following example shows how to enter the same VSC command from the DCL command line:

```
$ VSC SHOW FACILITY SYS
Facility SYS             "SYS Facility"
$
```

QUALIFIER

/INTERFACE=keyword

The /INTERFACE qualifier allows the user to select a keyword representing the prefered user interface presented by VSC. With the VSC/INTERFACE=DECWINDOWS keyword, the Motif graphical user interface is used. With the VSC/INTERFACE=CHARACTER_CELL keyword, the traditional command interface is used.

/KEPT_PROCESS (default)
/NOKEPT_PROCESS

Specifies whether VSC runs with or without creating a kept process. /KEPT_PROCESS specifies that the VSC utility should run in a kept subprocess that is retained from one invocation of the VSC command to the next. Use of a kept subprocess allows VSC to connect to the VSC database once for many invocations of VSC, thus speeding up subsequent accesses to the database. It also allows VSC to retain state information, such as the default stream and facility, between invocations. /NOKEPT_ PROCESS specifies that VSC not run in a kept subprocess.

Do *not* attempt to use the DCL ATTACH command to attach to a VDE or VSC kept process.

/LIBRARY=directory-spec

Specifies the mnemonic name or root directory of an alternate VDE library (other than the OpenVMS VAX library) with which you want to work. Subsequent VSC operations will access the database associated with that library.

EXAMPLE

```
$ VSC
VSC> SHOW FACILITY BAD
Facility BAD              "BAD Facility"
VSC> SHOW STREAM V5.4
Stream V5.4             "V5.4 SSB release (AETNA)"
VSC> EXIT
$
```

In this example the user enters the VSC command from DCL. VSC gives the user the VSC> prompt and the user enters two commands, SHOW FACILITY and SHOW STREAM. The EXIT command ends the VSC session and returns the user to the DCL prompt ($). The VSC

kept subprocess remains and will be reentered by each subsequent VSC command.

```
$ VSC SHOW FACILITY BAD; SHOW STREAM V5.4
Facility BAD              "BAD Facility"
Stream V5.4             "V5.4 SSB release (AETNA)"
$
```

This example shows the same two VSC commands entered directly on the DCL-level VSC command line. Both commands are executed, after which VSC exits and returns to the DCL prompt ($). In this case, VSC does not prompt for commands. Again, the kept subprocess remains.

3 VDE Command Reference

@

Executes a VDE command procedure.

FORMAT

@ *file-name*

PARAMETER

file-name

Indicates the name of the file from which VDE commands should be read. If no file extension is specified, the file type is assumed to be .VDE.

DESCRIPTION

A command procedure is a file containing a sequence of VDE commands. When the command procedure is executed, the commands in the file are processed in order as if each command had been entered individually from the terminal or workstation.

DCL Command Procedures: If invoking VDE commands from within a DCL command procedure, do not use a VDE kept subprocess.

EXAMPLES

1 `VDE> @INFILE`

This example shows how to execute the VDE commands in the file INFILE.VDE in your default directory. As there was no file extension explicitly specified in the example command, VDE defaults the file extension to .VDE.

2 `VDE> @[MYDIR]MYFILE.TMP`

This example shows how to execute the VDE commands in file MYFILE.TMP in directory [MYDIR].

ACCEPT REPLACEMENT

Marks a queued replacement as accepted in the VDE database. You can only mark a replacement as accepted if you are a reviewer for that replacement.

Requires RESREP privilege.

FORMAT

ACCEPT REPLACEMENT *rep-name [, rep-name...] ["remark"]*

PARAMETER

rep-name

The name of a queued replacement to be marked as accepted. You can accept more than one replacement at a time by using wildcard characters in the replacement name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Those queued replacements whose names match the wildcard pattern are marked as accepted.

"remark"

Specifies a one-line remark for the replacement. Typically this remark summarizes why the replacement is acceptable. Specifying the **remark** parameter is equivalent to specifying the /REMARK qualifier after the ACCEPT REPLACEMENT keywords.

DESCRIPTION

The ACCEPT REPLACEMENT command is used during the (optional) code review process supported by VDE, and is the negation of the REJECT REPLACEMENT command.

The ACCEPT REPLACEMENT command updates the VDE database to record that you have reviewed the code changes in a specified queued replacement and that you judge those changes to be acceptable for replacement into the VDE library—the replacement into the VDE library is performed with a subsequent PERFORM REPLACMENT command. You must be a reviewer for a replacement in order to issue to ACCEPT REPLACEMENT and mark the replacement as accepted.

The ACCEPT REPLACEMENT command allows you to attach a comment file to the queued replacement. This file gives your comments to the author of the replacement and to the other reviewers. If you have already created such a file, use the /INPUT qualifier to copy it to the replacement's staging area. If you want to create such a file, use the /EDIT qualifier. This qualifier causes VDE to invoke your preferred editor to create a comment file for the replacement. If you have already created a comment file for the replacement with an earlier ACCEPT or REJECT command, the /EDIT qualifier lets you edit that file again. This capability is useful if you change your mind or want to add more information.

ACCEPT REPLACEMENT

If you wish to mark a replacement as not acceptable, use the REJECT REPLACEMENT command. If you wish to see the review status of a queued replacement, see the documentation on the SHOW REPLACEMENT command in SHOW REPLACEMENT.

QUALIFIERS

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each replacement accepted. The /CONFIRM qualifier causes VDE to print a message for each replacement asking whether you want to accept that replacement. If you answer YES (or Y), that replacement is marked as accepted. If you answer NO (or N), the replacement status is not changed. If you enter a replacement name with a wildcard specification, VDE asks you to confirm the acceptance of each replacement whose name matches the wildcard specification. The /NOCONFIRM qualifier causes VDE to mark the specified replacements as accepted without asking for confirmation.

/EDIT

Specifies that VDE should invoke your preferred editor so that you can edit your comment file for each queued replacement. If such a comment file already exists (due to a previous ACCEPT or REJECT REPLACEMENT command for the same replacement), the editor modifies that file. Otherwise, the editor creates a new file in the replacement's staging directory. Its file name is your OpenVMS username and its file type is VDE$COMMENT. The comment file typically contains comments for the author of the replacement and the other reviewers.

This qualifier is ignored if you are running in batch mode.

/INPUT=file-spec

Specifies a comment file for the queued replacement. VDE reads the text file given by the **file-spec** parameter and writes its contents into a new file in the replacement's staging directory. The file name of the new file is your OpenVMS username and its file extension is VDE$COMMENT. The comment file typically contains remarks to the author of the replacement or information for the other reviewers.

/LOG (default)
/NOLOG

Controls whether log messages are printed after each replacement has been marked as accepted. The /LOG qualifier causes such messages to be printed and the /NOLOG qualifier suppresses them. These messages indicate that the replacement has been accepted and that the database transaction has successfully committed.

/REMARK="string"

Specifies a one-line remark for the replacement. Typically this remark summarizes why the replacement is acceptable. For longer comments, you may create a comment file with the /INPUT or /EDIT qualifier.

/USERNAME=username

Specifies that the replacement be marked as accepted on behalf of another user. The **username** parameter specifies the OpenVMS username of the other user. That user must be a reviewer for the specified replacement. You must have the USERNAME privilege to use this qualifier.

EXAMPLES

1

```
VDE> ACCEPT REPLACEMENT BROWN-3
%VDE-I-REPACCEPT, replacement BROWN-3 marked as accepted in database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example marks that the current user finds queued replacement BROWN–3 acceptable to be replaced into the VDE library.

2

```
VDE> ACCEPT REPLACEMENT SMITH-1/INPUT=COMMENT_FILE.TXT
%VDE-I-REPACCEPT, replacement SMITH-1 marked as accepted in database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

In this example, the user marks replacement SMITH–1 as accepted and provides a comment file that gives additional information to the user who queued the replacement.

ATTACH

Transfers control from your current process to the specified process.

FORMAT

ATTACH *process-name*

PARAMETER

process-name

Specifies the parent process or a spawned subprocess to which control is to be passed. The process must already exist, be part of your current job, and share the same input stream as your current process. However, the process cannot be the current process or a subprocess created with the /NOWAIT qualifier.

Do not specify the **process-name** parameter if you use the /IDENTIFICATION qualifier.

DESCRIPTION

The ATTACH command allows you to quickly transfer control between processes. (For information about creating a subprocess, see the SPAWN command.) For example, you can transfer control from a parent process to a subprocess using the ATTACH command. Then, in the subprocess, you can start a build job using VDE and, after you finish, you can enter the ATTACH command to move back to the parent process. If you later want to display the status of your build job, you can enter another ATTACH command at DCL level (or from many utilities) to move back to the VDE subprocess you created.

Do not attempt to use the DCL ATTACH command to attach to a VDE kept subprocess.

QUALIFIERS

/IDENTIFICATION=pid

Specifies the process identification (PID) of the process to which terminal control is transferred. Leading zeros can be omitted. Do not specify the **process-name** parameter when you use this qualifier.

EXAMPLES

1

```
$ EDIT VACATION.TXT
   .
   .
   .
CTRL/Y
$ SPAWN VDE
%DCL-S-SPAWNED, process SMITH_1 spawned
%DCL-S-ATTACHED, terminal now attached to process SMITH_1
VDE> BUILD GAME_PROG
VDE> ATTACH SMITH
%DCL-S-RETURNED, control returned to process SMITH
   .
   .
   .
```

```
CTRL/Y
$ ATTACH SMITH_1
VDE>
```

This example shows how to spawn a subprocess (SMITH_1) to invoke VDE and use the ATTACH command to move between VDE (SMITH_1) and the DCL command level (SMITH). The ATTACH command allows you to transfer control between subprocesses.

Note: Use the SPAWN command to create a new process and the ATTACH command to connect to a process that already exists.

BUILD

Creates and optionally starts a build job to build the specified modules. The discrete steps performed during the build are derived from VDE scripts.

Requires BUILD privilege.

FORMAT

BUILD *mod-name [, mod-name...]*

PARAMETER

mod-name

Specifies a module or build group to be produced by the build job.

If the parameter is a module name, it is typically the name of an executable image or another type of derived module that is the end result of a build operation. Each such module is a *target* of the build.

The module name must consist of a module name and a module type separated by a period (as in MODNAM.EXE) and may optionally be preceded by a facility name enclosed in square brackets (for example, [FACIL]MODNAM.EXE). If no facility name is specified, the module is assumed to belong to the default facility.

You can build more than one target module by using wildcard characters in any of the three components of the module name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. VDE builds those modules whose names match the wildcard pattern.

If the **mod-name** parameter specifies a build group, the BUILD command builds all modules that are members of the build group. A build-group name is a simple name that has been declared with a CREATE GROUP/BUILD command. A build-group name may not contain wildcard characters.

DESCRIPTION

The BUILD command performs two distinct operations: it creates a build job for the default development stream in the VDE database and it optionally starts that build job on a set of OpenVMS batch queues.

Creating a Build Job

To create a build job in the database, the BUILD command does the following three things:

- First, the BUILD command determines the target modules to be built. The target modules are specified by the **mod-name** parameters of the command.
- Next, the command determines the build steps that must be performed to build those target modules. The needed build steps are determined by the build dependencies stored in the VDE database, by the modules that have changed since the last time the targets were built, and by whether the user requests a full or minimal build.

- Finally, the BUILD command records information about the new build job in the database. The information specifies the build steps to be performed, any constraints on the order in which the steps are performed, and other attributes. (An example of an order constraint is the requirement that a compile step must complete before a dependent link step starts.)

The build job is always created for the default development stream. It uses the build dependencies of the default stream to determine the build steps to perform and it uses module replacement data for the default stream to determine the modules that have changed since the last build. The build job is then identified by the stream name and a build-job identification number. For example, the second build job for stream V1.0 is labeled ''build job 2 for stream V1.0'' in VDE messages.

You cannot create a new build job for a given stream if a build job already exists for that stream and that existing build job has not yet completed execution. This restriction ensures that two build jobs for the same stream do not execute at the same time. If you do not want the existing build job to run, stop it with the STOP BUILD_JOB command before creating the new build job.

Starting a Build Job

To start a build job, the BUILD command submits one or more batch jobs to the OpenVMS batch queues that the user specified either with the /QUEUE qualifier or by default. Then each batch job runs the VDE$BUILD utility that executes the build steps belonging to the build job. The steps are executed in a loop until all the steps are done. The VDE$BUILD utility does the following for each build step:

- The program looks in the database for the next available build step. (A build step, normally a compile, copy, or link step, is available if it is not waiting for another step to complete.) If no step is available, the program waits a few seconds before looking for another available build step.
- When the program finds a build step to run, it looks up the VDE script for that step and, from the script, generates the DCL commands needed to run the step. The program then sends the DCL commands to a subprocess that executes the build step, and waits for the step to complete.
- After the step completes, the program records the completion status of the step in the database. A step completes either successfully or with errors. If the step completes successfully, the program updates the step's build dependency information in the database and then looks for another step to perform.

The BUILD command always creates a new build job in the VDE database. It also starts the build job unless you use the /NOSTART qualifier or you answer NO to a query asking whether to start the build. If the BUILD command does not start the build job, you can start it later with the START BUILD_JOB command. After the build job is created (whether or not you start it), you can use the SHOW BUILD_JOB command to display

the attributes of the build, including the size of the build job and the exact build steps and build-step dependencies that make up the build job.

You can distribute the build job across several processors (whether in a single machine or in multiple machines in a cluster) by specifying the /QUEUE and /PROCESS_COUNT qualifiers. The /QUEUE qualifier specifies the batch queues to run the build job. The /PROCESS_COUNT qualifier specifies the total number of processes VDE submits to the batch queues performing the build job. By specifying the batch queues and the number of processes VDE submits to each queue, you can decide how to distribute the build job over a cluster with processors of different speeds and configurations.

If you do not specify the /QUEUE or /PROCESS_COUNT qualifier, VDE uses the default batch queues and process counts of the default development stream. You can establish these defaults using the /QUEUE and /PROCESS_COUNT qualifiers with the CREATE STREAM and MODIFY STREAM commands. If you have not specified such defaults, VDE submits one process to queue SYS$BATCH.

QUALIFIERS

/AFTER=date-time

Indicates the date and time at which the build job begins execution. Specify the date and time using OpenVMS date-time syntax. Use this qualifier only if the BUILD command starts the build job: the qualifier specifies the start time for each batch job that is submitted to execute the build job.

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want the build job started. The /CONFIRM qualifier causes VDE to print a message asking whether you want to start the build job. This message is displayed after VDE creates the build job and displays a message stating the size of the job. If you answer YES (or Y), the build job is started. If you answer NO (or N), the build job is not started. The /NOCONFIRM qualifier causes VDE to start the build job without asking for confirmation. Neither qualifier has any effect if you use the /NOSTART qualifier: /NOSTART unconditionally prevents VDE from starting the build job.

/FULL

Directs the BUILD command to perform a full build of your software system. In a full build, all source modules that contribute to the target modules you specified are compiled or otherwise processed. If you do not specify the /FULL qualifier, the BUILD command performs a minimal system build, which means that only the source modules that have changed since the last build of the specified target modules are compiled or processed. A minimal build is normally much quicker than a full build because it performs only the minimal work needed to propagate your source changes to the target modules.

/LOG (default)
/NOLOG

Controls whether log messages are printed after the build job is created and submitted to batch queues for execution. The /LOG qualifier causes

such messages to be printed and the /NOLOG qualifier suppresses them. These messages indicate that the build job has been created, that the database transaction has successfully committed, and that batch jobs have been submitted to various batch queues to execute the build job.

/PROCESS_COUNT=proc-count

Specifies the number of processes to execute the build job. VDE creates the processes by submitting batch jobs to the queues specified by the /QUEUE qualifier or to the default queues for the default stream. The number of processes that VDE creates (that is, the number of OpenVMS batch jobs it submits) is specified by the **proc-count** parameter. As a result, this parameter determines the degree of parallelism for the build job.

/QUEUE=(queue-name [,queue-name...])

Specifies the OpenVMS batch queues to which the build job is submitted. Each **queue-name** parameter specifies the name of one batch queue. The **queue-name** parameter may optionally be followed by a process count in parentheses. This count specifies the number of processes that are started on the queue to run the build job. For example, SYS$BATCH(3) causes three build-job processes to be submitted to queue SYS$BATCH. If the process count is omitted, one process is submitted to the queue.

If you only specify one **queue-name** parameter, the parentheses may be omitted.

If you also use the /PROCESS_COUNT qualifier, the /PROCESS_COUNT qualifier states the total number of build-job processes to create. In that case, the process counts attached to the **queue-name** parameters on the /QUEUE qualifier are used as scaling factors to distribute build-job processes among the queues proportionally. For example, if you specify queues HI_BATCH, HO_BATCH(3) and a total process count of eight, then two processes are submitted to queue HI_BATCH and six processes to queue HO_BATCH. If the total process count does not divide evenly into the sum of the scaling factors, the remaining processes are allocated to the queues in the order the queues are listed. If the total process count is nine, for example, the one extra process is allocated to queue HI_BATCH because HI_BATCH is listed first.

/REMARK="string"

Specifies a remark string to be stored in the database for the new build job. The quoted remark string, which can be up to 132 characters long, may contain any explanatory remark about the build job that you want to store in the database.

/START (default)
/NOSTART

Controls whether the build job is started or not. The /START qualifier starts the build job, which means that VDE submits batch jobs to the appropriate OpenVMS batch queues. These batch jobs run programs that execute all the build steps that make up the build job. The /NOSTART qualifier prevents VDE from submitting the batch jobs. If you use the /NOSTART qualifier, you can start the build job later with the START BUILD_JOB command.

/STEP_DELAY=time-interval

Specifies the step-delay time interval used by VDE$BUILD. VDE$BUILD is the program that executes the build steps belonging to a build job. If VDE$BUILD cannot find an available build step in the database, the program waits the amount of time specified by the /STEP_DELAY qualifier before it looks in the database again.

When multiple processes run a build job, one process may be idle until a second process completes its current build step. In that situation, the first process periodically looks in the database to see if the second process has finished its step. The step-delay interval determines how often the first process looks in the database for an available step.

Use the following syntax to denote the number of hours, minutes, and seconds in the interval:

```
/STEP_DELAY=hh:mm:ss.ss
```

For example, an interval of three and one half seconds is specified as follows:

```
/STEP_DELAY=00:00:03.50
```

Usually the interval is set at a few seconds. If this qualifier is not specified, VDE uses the default step-delay interval of the stream. If no default has been specified, the delay interval is five seconds.

/TIMEOUT=time-interval

Specifies the timeout interval for the build job. If no build steps complete during this interval, the VDE$BUILD utility stops the build job and assigns it TIMEOUT status. Therefore, the timeout interval must be longer than the time required to complete the longest build step.

Normally, a build job only times out if a node that is running one of the build-job processes fails. When a node fails, other build-job processes may be unable to proceed because they are waiting for a build step that ran on the failed node to complete, which will not happen. In that case, after the timeout interval expires, one of the waiting processes marks the build job as timed out, after which the remaining processes terminate themselves. You can later restart the build job using the START BUILD_JOB command.

Use the following syntax to denote the number of hours, minutes, and seconds in the interval:

```
/TIMEOUT=hh:mm:ss.ss
```

For example, an interval of two and one half hours is specified as follows:

```
/TIMEOUT=02:30:00.00
```

The length of the timeout interval should be set based on the length of the longest build step in the build job. If this qualifier is not specified, VDE uses the default timeout interval of the stream. If no default has been specified, the timeout interval is one hour.

EXAMPLES

1

```
VDE> BUILD FOO.EXE
%VDE-I-BLDJOBCRE, build job 1 for stream BL22 has been created
%VDE-I-BLDJOBSIZ, build job 1 for stream BL22 consists of 5 steps
%VDE-I-COMMIT, database transaction has successfully committed
%VDE-I-BLDJOBSTARTING, starting build job 1 for stream BL22
%VDE-I-BLDJOBENT, entry number 308 submitted to queue SYS$BATCH
   Job VDEBUILD_0001 (queue CLUSTER_BATCH, entry 308) started on PHI_BATCH
%VDE-I-BLDJOBSTARTED, build job 1 for stream BL22 started with 1 processes
VDE>
```

This example performs a minimal system build to build module FOO.EXE. The created build job is build job 1 for stream BL22. The build job consists of five build steps, such as compile and link steps. Because the /NOSTART qualifier is omitted, the build job is started on the default queue for the stream, SYS$BATCH. The build job runs on only one process.

2

```
VDE> BUILD/FULL/NOSTART BLDGRP
%VDE-I-BLDJOBCRE, build job 2 for stream BL22 has been created
%VDE-I-BLDJOBSIZ, build job 2 for stream BL22 consists of 125 steps
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

In this example, build job 2 for the default stream, BL22, is created. This is a full build (not a minimal build) that builds all modules that are members of build group BLDGRP. This build requires 125 build steps. The build job is not started because the /NOSTART qualifier is specified. (To start it later, use the START BUILD_JOB command.) Although the build job is not started, it exists in the database so you can display information about it using the SHOW BUILD_JOB command.

3

```
VDE> BUILD/PROCESS_COUNT=3 *.EXE
%VDE-I-BLDJOBCRE, build job 2 for stream V2.0-3 has been created
%VDE-I-BLDJOBSIZ, build job 2 for stream V2.0-3 consists of 25 steps
%VDE-I-COMMIT, database transaction has successfully committed
%VDE-I-BLDJOBSTARTING, starting build job 2 for stream V2.0-3
%VDE-I-BLDJOBENT, entry number 317 submitted to queue SYS$BATCH
   Job VDEBUILD_0001 (queue CLUSTER_BATCH, entry 317) started on PHI_BATCH
%VDE-I-BLDJOBENT, entry number 318 submitted to queue SYS$BATCH
   Job VDEBUILD_0001 (queue CLUSTER_BATCH, entry 318) started on FOO_BATCH
%VDE-I-BLDJOBENT, entry number 319 submitted to queue SYS$BATCH
   Job VDEBUILD_0001 (queue CLUSTER_BATCH, entry 319) started on FUM_BATCH
%VDE-I-BLDJOBSTARTED, build job 2 for stream V2.0-3 started with 3 processes
VDE>
```

This example creates and starts a build job that builds all executable images for the default stream. It runs the build job with three parallel processes, all submitted to queue SYS$BATCH. The batch jobs execute on three separate batch queues, in this case on separate nodes in a cluster.

4
```
VDE> BUILD/QUEUE=(PHI_BATCH,FOO_BATCH(2),FUM_BATCH) *.EXE
%VDE-I-BLDJOBCRE, build job 3 for stream V2.0-3 has been created
%VDE-I-BLDJOBSIZ, build job 3 for stream V2.0-3 consists of 25 steps
%VDE-I-COMMIT, database transaction has successfully committed
%VDE-I-BLDJOBSTARTING, starting build job 3 for stream V2.0-3
%VDE-I-BLDJOBENT, entry number 417 submitted to queue PHI_BATCH
   Job VDEBUILD_0001 (queue PHI_BATCH, entry 417) started on PHI_BATCH
%VDE-I-BLDJOBENT, entry number 418 submitted to queue FOO_BATCH
   Job VDEBUILD_0001 (queue FOO_BATCH, entry 418) started on FOO_BATCH
%VDE-I-BLDJOBENT, entry number 419 submitted to queue FOO_BATCH
   Job VDEBUILD_0001 (queue FOO_BATCH, entry 419) started on FOO_BATCH
%VDE-I-BLDJOBENT, entry number 420 submitted to queue FUM_BATCH
   Job VDEBUILD_0001 (queue FUM_BATCH, entry 420) started on FUM_BATCH
%VDE-I-BLDJOBSTARTED, build job 3 for stream V2.0-3 started with 4 processes
VDE>
```

Like the previous example, this example creates and starts a build job that builds all executable images. In this case, the /QUEUE qualifier specifies that four batch jobs are submitted: one to batch queue PHI_BATCH, two to queue FOO_BATCH, and one to queue FUM_BATCH. These queues belong to specific nodes in the cluster. As a result, the /QUEUE qualifier determines precisely how the build job is distributed across the nodes of the cluster.

5
```
VDE> BUILD/CONFIRM FOO.EXE
%VDE-I-BLDJOBCRE, build job 8 for stream BL22 has been created
%VDE-I-BLDJOBSIZ, build job 8 for stream BL22 consists of 2 steps
%VDE-I-COMMIT, database transaction has successfully committed
Start build job 8 for stream BL22 ? [No]: YES
%VDE-I-BLDJOBSTARTING, starting build job 8 for stream BL22
%VDE-I-BLDJOBENT, entry number 79 submitted to queue SYS$BATCH
   Job VDEBUILD_0001 (queue CLUSTER_BATCH, entry 79) started on PHI_BATCH
%VDE-I-BLDJOBSTARTED, build job 8 for stream BL22 started with 1 processes
VDE>
```

The /CONFIRM qualifier causes VDE to ask the user whether to start the new build job. VDE asks the question after creating the build job and after displaying its size (in this example, only two steps). The user answers YES and VDE starts one process for the build job on queue SYS$BATCH.

CANCEL DIRECTORY

Cancels a directory template in the VDE database. Directory templates determine what disk directories VDE creates when creating new development streams and facilities.

Requires CRESTRM privilege.

FORMAT CANCEL DIRECTORY

DESCRIPTION

The CANCEL DIRECTORY command deletes a ''directory template'' from the VDE database. Directory templates are defined with the SET DIRECTORY command and specify the names that VDE should assign to disk directories when creating directories for new development streams or facilities or for the library's queued-replacement staging area. The CANCEL DIRECTORY command can delete all templates that the SET DIRECTORY command can create. The CANCEL DIRECTORY command does not delete or otherwise affect existing disk directories, however.

Unless the cancelled template is for the library's delta files or its staging area, it is only cancelled for the default development stream. Because directory templates are inherited by a stream's descendants, the CANCEL DIRECTORY command will affect the disk directories created for all future descendants of the default stream. It may also affect the directories created for new facilities in the default stream.

When you have cancelled a directory template, VDE uses its default rules to create directory trees for subsequent streams and facilities. VDE always creates stream, facility, and delta-file root directories, whether there are templates for such directories or not. VDE does not create facility subdirectories unless there are templates for them, however, so if you cancel a subdirectory template, you prevent that subdirectory from being created for new streams or facilities.

For additional information on directory creation, and on directory symbols (*dir-symbol*) and directory symbol processing, see SET DIRECTORY, SHOW DIRECTORY, and Script Functions.

QUALIFIERS

/DELTA_FILES
/STAGING
/STREAM (default)

Specifies whether a directory template for delta files, the staging area, or the default stream is deleted. The /DELTA_FILES qualifier specifies that a directory template for the VDE library's delta files (CMS libraries) be deleted. The /STAGING qualifier specifies that the directory template for the library's staging-area root directory be deleted. (Staging areas hold files for queued replacements.) The /STREAM qualifier specifies that a directory template for the default stream be deleted.

Of /DELTA_FILES, /STAGING and /STREAM, the latter is the default.

/FACILITY=fac-name

Specifies that a directory template for the facility specified by the **fac-name** parameter be deleted. If both this qualifier and the /SUBDIRECTORY qualifier are specified, VDE deletes the directory template for the specified facility subdirectory. If the /FACILITY qualifier is specified and the /SUBDIRECTORY qualifier is omitted, VDE deletes the directory template for the specified facility's root directory.

If the /FACILITY qualifier is omitted and the /SUBDIRECTORY qualifier is specified, VDE deletes the directory template for the given facility subdirectory for all facilities that are not explicitly specified in other templates. If both the /FACILITY qualifier and the /SUBDIRECTORY qualifier are omitted, VDE deletes the template for the stream or delta-file root directory.

You can delete more than one directory template at a time by using wildcard characters in the **fac-name** parameter. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Directory templates for those facilities whose names match the wildcard pattern are deleted.

/LOG (default)
/NOLOG

Controls whether log messages are printed after the directory template is deleted. The /LOG qualifier causes such messages to be printed and the /NOLOG qualifier suppresses them. These messages indicate that the directory template has been deleted from the database and that the database transaction has completed successfully.

/SUBDIRECTORY=dir-symbol

Specifies that a directory template for a facility subdirectory be deleted. The **dir-symbol** parameter specifies the directory symbol that identifies the subdirectory. Among other uses, the directory symbol is used within VDE scripts; see Script Keywords for further details. If this qualifier is omitted, VDE deletes the directory template for a facility, stream, or delta-file root directory. (The description of the /FACILITY qualifier explains how the qualifiers interact to select the template to delete.)

EXAMPLES

1

```
VDE> CANCEL DIRECTORY/FACILITY=RMS
%VDE-I-DIRDEL, directory specification [MYDIR.RMS] deleted from database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example cancels the directory template for the root directory for facility RMS. The log messages show that the template was successfully deleted from the database.

2

```
VDE> CANCEL DIRECTORY/SUBDIRECTORY=ANASYM
%VDE-I-DIRDEL, directory specification [.ANA] deleted from database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example cancels the directory template for the subdirectory with directory symbol ANASYM. VDE deletes the subdirectory template that is used for all facilities that do not have explicit subdirectory templates.

CANCEL FOLD

Deletes one or more fold records from the VDE database. A fold record is a reminder that a source change should be manually "folded" into a given development stream in order to propagate the change.

Requires RESREP privilege.

FORMAT

CANCEL FOLD *mod-name [, mod-name...]*

PARAMETER

mod-name

Specifies a module for which a fold record is to be deleted. The module name consists of an optional facility name enclosed in square brackets, a module name, and an optional type name preceded by a period (such as [FACIL]MOD1.MAR). If the facility name is omitted, the module is assumed to belong to the default facility. If the type name is omitted, VDE deletes fold records for all source modules with the specified module name in the given facility.

You can delete fold records for more than one module by using wildcard characters in any of the three components of the module name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. VDE deletes fold records for all source modules whose names match the wildcard pattern.

DESCRIPTION

A fold record is a reminder that a source change should be manually folded into a given development stream in order to propagate the change into that stream. The CANCEL FOLD command deletes each specified fold record from the VDE database. This command is a synonym for the DELETE FOLD command.

QUALIFIERS

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each fold record deleted. The /CONFIRM qualifier causes VDE to print a message for each fold record asking whether you want that record deleted. If you answer YES (or Y), that fold record is deleted from the database. If you answer NO (or N), the fold record is not deleted. If you enter a module name with a wildcard specification, VDE asks you to confirm the deletion of fold records for each module whose name matches the wildcard specification. The /NOCONFIRM qualifier causes VDE to delete the specified fold records without asking for confirmation.

/IDENTIFICATION=fold-ident

Specifies the fold record to be deleted. This qualifier is required when you have multiple fold records for the same module in the same stream. The **fold-ident** parameter is the fold identifier of the fold record to be deleted.

The fold identifier is the identifier you specified or VDE assigned when you created the fold record, usually a small integer value. Use the SHOW FOLD command to determine the fold identifier of each fold record.

/LOG (default)
/NOLOG

Controls whether log messages are printed after each fold record has been deleted. The /LOG qualifier causes such messages to be printed and the /NOLOG qualifier suppresses them. These messages indicate that the fold record has been deleted and that the database transaction has successfully committed.

/STREAM=stream-name

Specifies the development stream for which fold records should be deleted. The **stream-name** parameter gives the name of the stream. If this qualifier is omitted, VDE deletes fold records for the default development stream only.

/USERNAME=username

Specifies that fold records are to be deleted on behalf of another user. The **username** parameter specifies the OpenVMS username of the other user. You must have the USERNAME privilege to use this qualifier.

EXAMPLES

1

```
VDE> CANCEL FOLD [CODE]X.FOR/IDENT=1
%VDE-I-FOLDDEL, fold record 1 for module [CODE]X.FOR has been deleted
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example deletes fold record 1 for module [CODE]X.FOR. The log messages confirm that the fold record has been successfully deleted.

CANCEL LOG

Disables the recording of VDE input and output in a log file. Input and output logging is enabled using the SET LOG command.

FORMAT CANCEL LOG

PARAMETERS *None.*

QUALIFIERS *None.*

EXAMPLES

1
```
VDE> CANCEL LOG
VDE> SHOW LOG
No output logging
VDE>
```

The CANCEL LOG command disables input and output logging. The SHOW LOG command confirms that logging is disabled.

CANCEL PRIVILEGES

Disables VDE privileges for the current VDE session.

FORMAT

CANCEL PRIVILEGES *priv [, priv...]*

PARAMETER

priv

The name of a VDE privilege to disable for the current VDE session.

For a list of privileges, see Table 3–5 in SET PRIVILEGES.

DESCRIPTION

The VDE privileges enabled for your current VDE session determine the VDE commands (and in some cases command qualifiers) you are allowed to use. Use the CANCEL PRIVILEGES command to disable privileges for the current VDE session. You may want to use the command to prevent accidental changes to the library database, for example.

You can reestablish privileges using the SET PRIVILEGES command and you can display your current privileges using the SHOW PRIVILEGES command.

QUALIFIERS

None.

EXAMPLES

1

```
VDE> CANCEL PRIVILEGES FROZEN, USERNAME
VDE> SHOW PRIVILEGES
Currently set privileges:
      BUILD CREFAC CREGRP CREMOD CRESTEP RESREP
VDE>
```

The CANCEL PRIVILEGES command disables the FROZEN and USERNAME privileges. THE SHOW PRIVILEGES command confirms that those privileges are no longer set.

CANCEL REPLACEMENT

This command is the negation of the PERFORM REPLACEMENT command. This command allows a project leader or other reviewer to cancel one or more accepted, queued replacements, and thus prevent these modules from being relocated into the VDE library via a PERFORM REPLACEMENT command.

Requires RESREP privilege to cancel your own queued replacement. Requires PERFREP privilege to cancel another user's queued replacement.

FORMAT

CANCEL REPLACEMENT *rep-name [, rep-name...]*

PARAMETER

rep-name

The name of a queued replacement to be cancelled. You can cancel more than one replacement at a time by using wildcard characters in the replacement name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Those queued replacements whose names match the wildcard pattern are cancelled.

DESCRIPTION

The CANCEL REPLACEMENT command is used after the module(s) in the replacement have been replaced by a user and have been accepted through the (optional) code review process supported by VDE. Cancellation prevents these modules from being moved into the VDE library. This operation is the negation of the PERFORM REPLACEMENT command.

The CANCEL REPLACEMENT command cancels each specified queued replacement so that the replacement is deleted from the VDE library and is no longer eligible to be performed by the PERFORM REPLACEMENT command.

The modules belonging to the cancelled replacement will remain reserved to the user that originally reserved and replaced them, and that user can later REPLACE updated modules, or can UNRESERVE the modules to entirely cancel the outstanding reservation(s). If subsequently replaced, the modules will again have to be accepted (or rejected) by the reviewers.

For information on how a queued replacement is accepted or rejected, see the ACCEPT REPLACEMENT and REJECT REPLACEMENT commands.

QUALIFIERS

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each replacement cancelled. The /CONFIRM qualifier causes VDE to print a message for each replacement asking whether you want that replacement cancelled. If you answer YES (or Y), that replacement is cancelled. If you answer NO (or N), the replacement is not cancelled. If you enter a

replacement name with a wildcard specification, VDE asks you to confirm the cancellation of each replacement whose name matches the wildcard specification. The /NOCONFIRM qualifier causes VDE to cancel the specified replacements without asking for confirmation.

/LOG (default)
/NOLOG

Controls whether log messages are printed after each replacement has been cancelled. The /LOG qualifier causes such messages to be printed and the /NOLOG qualifier suppresses them. These messages indicate that the replacement has been cancelled and that the database transaction has successfully committed.

EXAMPLES

1
```
VDE> CANCEL REPLACEMENT SMITH-2
%VDE-I-REPCAN, replacement SMITH-2 cancelled
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example cancels replacement SMITH–2 so that this replacement is no longer queued in the VDE database. The modules associated with the replacement are still reserved after the replacement is cancelled.

CANCEL SESSION

Cancels one or more reservation sessions by deleting them from the VDE database.

Requires RESREP privilege to cancel your own session. Requires PERFREP privilege to cancel another user's session.

FORMAT

CANCEL SESSION *session-name [, session-name...]*

PARAMETER

session-name

The name of a reservation session to be cancelled. You can cancel more than one session at a time by using wildcard characters in the session name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Those reservation sessions whose names match the wildcard pattern are cancelled.

DESCRIPTION

The CANCEL SESSION command cancels each specified reservation session so that the session is deleted from the VDE database. The reservations that belonged to the session remain in the database, but no longer belong to the cancelled session.

If you want to both delete a session and unreserve all modules associated with the session, use the UNRESERVE command with the /SESSION qualifier.

QUALIFIERS

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each session cancelled. The /CONFIRM qualifier causes VDE to print a message for each session asking whether you want that session cancelled. If you answer YES (or Y), that session is cancelled. If you answer NO (or N), the session is not cancelled. If you enter a session name with a wildcard specification, VDE asks you to confirm the cancellation of each session whose name matches the wildcard specification. The /NOCONFIRM qualifier causes VDE to cancel the specified sessions without asking for confirmation.

/LOG (default)
/NOLOG

Controls whether log messages are printed after each session has been cancelled. The /LOG qualifier causes such messages to be printed and the /NOLOG qualifier suppresses them. These messages indicate that the session has been cancelled and that the database transaction has successfully committed.

EXAMPLES

1

```
VDE> CANCEL SESSION SMITH-3
%VDE-I-SESCAN, session SMITH-3 cancelled
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example cancels reservation session SMITH–3 so that this session is no longer recorded in the VDE database. The modules associated with the session are still reserved after the session is cancelled.

CANCEL VERIFY

Cancels display of command input to the terminal.

FORMAT CANCEL VERIFY

PARAMETERS

None.

DESCRIPTION

The CANCEL VERIFY command disables the echoing of command input to the terminal. It cancels the effect of the SET VERIFY command. The CANCEL VERIFY command normally is used to disable the echoing of input commands from command procedures.

The CANCEL VERIFY and SET NOVERIFY commands are equivalent.

QUALIFIERS

None.

EXAMPLES

1

```
VDE> SET VERIFY
VDE> SHOW MODULE MOD1.FOR
SHOW MODULE MOD1.FOR
Module [FACIL]MOD1.FOR
VDE> CANCEL VERIFY
CANCEL VERIFY
VDE> SHOW FACILITY FACIL
Facility FACIL
VDE>
```

The SET VERIFY command allows the terminal to display (echo) the command line it receives. The SHOW MODULE command is therefore echoed on the terminal before the output of the command is displayed. The CANCEL VERIFY command is also echoed but it disables the display of subsequent commands. The SHOW FACILITY command that follows is not displayed on the terminal but the output of the command is.

COMPILE

Compiles the specified modules using the appropriate scripts from the database.

Requires BUILD privilege.

FORMAT

COMPILE *mod-name [, mod-name...]*

PARAMETER

mod-name

Specifies a module to be compiled. The module name consists of an optional facility name enclosed in square brackets, a module name, and an optional type name preceded by a period, such as [FACIL]MOD1.MAR. If no facility name is specified, the default facility is assumed. If no type name is specified, all source modules of the specified module name in the given facility are compiled.

You can compile more than one module by using wildcard characters in any of the three components of the module name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. VDE compiles those modules whose names match the wildcard pattern.

DESCRIPTION

The COMPILE command compiles a module the same way it is compiled in a build job. The COMPILE command retrieves the compile script for each specified module and generates a DCL command file from the script. The command then executes the command file in a subprocess. If the module compiles successfully, VDE updates the build dependency information for the compile step in the database.

The COMPILE command can be used to compile a module without starting a build job. It is often used to test compile scripts. It can also be used to collect the initial dependency information for a compile step that does not yet have dependency information in the database.

The COMPILE command only operates on modules that have a generation in the current stream (the default stream or the stream specified with the /STREAM qualifier). In addition, it only operates on source modules unless you specify the type name in the **mod-name** parameter without wildcard characters. In other words, unless you explicitly specify a derived-module file extension, the command assumes that you only want to compile source modules.

QUALIFIERS

/LOG (default)
/NOLOG

Controls whether log messages are printed after the compile step executes. The /LOG qualifier causes such messages to be printed and the /NOLOG qualifier suppresses them. These messages indicate that the compile step

has completed, either successfully or with errors, and that the dependency information for the step has been updated in the VDE database.

/STREAM=stream-name

Specifies that the modules be compiled for the stream given by the **stream-name** parameter. If this qualifier is omitted, the modules are compiled for the default stream.

EXAMPLES

1

```
VDE> COMPILE B.C
$ DEFINE/NOLOG SRC$ DEV$:[PROJDIR.MAIN.CODE.SRC]
$ DEFINE/NOLOG OBJ$ DEV$:[PROJDIR.MAIN.CODE.OBJ]
$ VDE FETCH [CODE]B.C/OUTPUT=SRC$:
%VDE-I-FETCHED, module generation [CODE]B.C;1(1) fetched from library
%VDE-I-OUTFILE, output file is DEV$:[PROJDIR.MAIN.CODE.SRC]B.C;
$ CC/ANA=OBJ$:/OBJECT=OBJ$:  SRC$:B.C
$ VDE/DEPEND C OBJ$:B.ANA
$ !
$ SET NOVERIFY   ! Successful script completion
%VDE-I-COMPSUCC, compilation of [CODE]B.C completed successfully
%VDE-I-DEPUPDATE, dependency information updated for build step
VDE>
```

This example compiles module B.C in the default facility, facility CODE. The output from the execution of the compile script is displayed on the terminal. The final log messages show that the compile step completed successfully and that its dependency information was updated in the database.

2

```
VDE> COMPILE [RMS]E.B32
$ VDE FETCH [RMS]E.B32/OUTPUT=DEV$:[PROJDIR.MAIN.RMS.SRC]
%VDE-I-FETCHED, module generation [RMS]E.B32;3(2A1) fetched from library
%VDE-I-OUTFILE, output file is DEV$:[PROJDIR.MAIN.RMS.SRC]E.B32;
$ DEFINE/NOLOG SRC$ DEV$:[PROJDIR.MAIN.RMS.SRC]
$ DEFINE/NOLOG OBJ$ DEV$:[PROJDIR.MAIN.RMS.OBJ]
$ BLISS/OBJECT=OBJ$:/ANA=OBJ$:  SRC$:E.B32
;   0067  2         RETRUN;
% WARN#000      ....1  L1:0067
  Undeclared name:  RETRUN
%BLS32-W-ENDDIAGS, Compilation completed with 1 diagnostic
$VDE$ERROR:
$ SET NOVERIFY   ! Script completed with errors
%VDE-I-COMPFAIL, compilation of [RMS]E.B32 completed with errors
VDE>
```

This example compiles module [RMS]E.B32 but the compilation terminates with errors. The output of the compile step includes error messages from the compiler, and the final log message shows that the step failed. In this case, VDE does not update the database dependency information.

CONVERT LIBRARY

Converts the current VDE library to the format required by the current version of VDE.

Requires MODLIB privilege if automatic conversion is disabled.

FORMAT CONVERT LIBRARY

PARAMETERS

None.

DESCRIPTION

When a new version of VDE is installed on your system, that new version frequently requires a different library format than the previous version. The new version may require a different database schema or new library subdirectories, for example. Because your existing VDE libraries are in the old library format, you must convert them to the new format. The CONVERT LIBRARY command performs this conversion.

The VDE database in your library contains a ''format version number'' that tells VDE what format your library has. If the current version of VDE requires a more recent library format, the CONVERT LIBRARY command converts the library and its database to that more recent format. It then updates the format version number in the database.

Once you have converted a VDE library to a more recent format, you can no longer use older versions of VDE on that library. If you want to retain the option of going back to an older version of VDE, you should back up your library and its database before your convert them.

If you have specified the /AUTO_CONVERT qualifier with the CREATE or MODIFY LIBRARY command, VDE will automatically convert your library to the new format the first time a user accesses the library database using the new version of VDE. Automatic conversion is more convenient than using the CONVERT LIBRARY command, but does not give you the option of backing up the library before converting it. If you have specified the /NOAUTO_CONVERT qualifier with the CREATE or MODIFY LIBRARY command, you must convert the library with an explicit CONVERT LIBRARY command. See the descriptions of the CREATE LIBRARY and CONVERT LIBRARY commands for a full explanation.

QUALIFIERS

None.

EXAMPLES

1

```
VDE> CONVERT LIBRARY
%VDE-I-LIBCONV, library format converted for new version of VDE
VDE>
```

This command converts the current library to the format required for the new version of VDE. The log message shows that the conversion successfully completed.

COPY

Copies the specified modules using the appropriate scripts from the database.

Requires BUILD privilege.

FORMAT

COPY *mod-name [, mod-name...]*

PARAMETER

mod-name

Specifies a module to be copied using a copy script. The module name consist of an optional facility name enclosed in square brackets, a module name, and an optional type name preceded by a period, such as [FACIL]MOD2.H. If no facility name is specified, the default facility is assumed. If no type name is specified, all modules of the specified module name in the given facility are copied.

You can copy more than one module by using wildcard characters in any of the three components of the module name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. VDE copies those modules whose names match the wildcard pattern.

DESCRIPTION

The COPY command copies a module the same way it is copied in a build job. The COPY command retrieves the copy script for each specified module and generates a DCL command file from the script. The command then executes the command file in a subprocess.

A copy step moves a module into the directory or other location where other build steps later expect to find it. For example, copy steps move INCLUDE files from delta files into clear-copy files in the subdirectories where other build steps will later access them. Unlike a compile or link step, a copy step does not transform one set of modules into a different set of modules; it only copies a single module from one form or location to another.

The COPY command can be used to perform a copy step without starting a build job. It is often used to test copy scripts.

The COPY command only operates on modules that have a generation in the current stream (the default stream or the stream specified with the /STREAM qualifier). In addition, it only operates on source modules unless you specify the type name in the **mod-name** parameter without wildcard characters. In other words, unless you explicitly specify a derived-module file extension, the command assumes that you only want to copy source modules.

QUALIFIERS

/LOG (default)
/NOLOG

Controls whether log messages are printed after the copy step executes. The /LOG qualifier causes such messages to be printed and the /NOLOG qualifier suppresses them. These messages indicate that the copy step has completed, either successfully or with errors, and that the dependency information for the step has not been updated in the VDE database. (Copy steps do not normally update dependency information since the only input and output to the step is the module being copied.)

/STREAM=stream-name

Specifies that the modules be copied for the stream given by the **stream-name** parameter. If this qualifier is omitted, the modules are copied for the default stream.

EXAMPLES

1

```
VDE> COPY [CODE]A.REQ
$ VDE FETCH [CODE]A.REQ/OUTPUT=DEV$:[PROJDIR.MAIN.CODE.SRC]
%VDE-I-FETCHED, module generation [CODE]A.REQ;1(1) fetched from library
%VDE-I-OUTFILE, output file is DEV$:[PROJDIR.MAIN.CODE.SRC]A.REQ;
$ !
$ SET NOVERIFY   ! Successful script completion
%VDE-I-COPYSUCC, copy of [CODE]A.REQ completed successfully
%VDE-I-DEPNOTUPDATE, dependency information not updated for build step
VDE>
```

This example performs the copy action for module [CODE]A.REQ. The copy script fetches the latest generation for module A.REQ from its delta file and leaves it in clear-copy form in the [.SRC] subdirectory for facility CODE and the default development stream. There it is available to future compile steps. The output from the copy script is displayed on the terminal as shown. The final log messages show that the copy step completed successfully and that its dependency information was not updated in the database.

CREATE ARCHITECTURE

Defines one or more architecture table entries in the VDE database. Each architecture table entry...

Requires the CREARCH privilege.

ARCHITECTURE support is not fully implemented.

An architecture is a construct that encompasses one or more facilities, and allows modules to exist in a "common" or "default" facility, and zero or more architecture specific facilities. In must the same fashion as the SYS$SYSROOT: logical name allows one to support both system-specific and cluster-common files, an architecture allows one to support architecture-specific and architecture-common modules.

FORMAT

CREATE ARCHITECTURE *arch-name [, arch-name...]*

PARAMETER

arch-name

The name of the architecture to be created. This name may be up to 39 characters long and must follow VDE name syntax.

DESCRIPTION

Under Construction...

QUALIFIERS

/DEFER (default)
/NODEFER

Under Construction...

/DELETE (default)
/NODELETE

Under Construction...

/LOG (default)
/NOLOG

Under Construction...

/PRIVATE (default)
/PUBLIC

Under Construction...

/REMARK="string"

Specifies a remark string to be stored for the new architecture. The quoted remark string, which can be up to 132 characters long, may contain any explanatory remark about the architecture. that you want stored in the VDE database. Typically the remark describes the function or purpose of the architecture.

EXAMPLES

1
```
VDE> CREATE ARCHITECTURE VAXPDP/REMARK="Version n"
VDE>
```

Under Construction...

CREATE CONTEXT

Defines one or more contexts in the VDE database. Each context can be used to specify the default development stream, default architecture and default facility for a VDE session, and one context can be used to set the default stream and facility when first connecting to a VDE database.

Requires the CRECTX privilege to create public contexts.

FORMAT

CREATE CONTEXT *ctx-name [, ctx-name...]*

PARAMETER

ctx-name

The name of the context to be created. This name may be up to 39 characters long and must follow VDE name syntax.

DESCRIPTION

A ''context'' specifies defaults for VDE sessions. When you create a context using the CREATE CONTEXT command, you specify the name of the context, the associated development stream, and the associated facility.

To apply the defaults associated with a context, specify the context name as a parameter to the SET CONTEXT command. To set a particular context as the default context for use when you issue the SET LIBRARY command, use the SET CONTEXT/DEFAULT command to mark a particular context as the default context.

There are two kinds of contexts available:

- the *private context* is available to and only visible to the user who created it.
- the *public context* is available to all users of the current VDE library.

You do not need special VDE privileges to create a private context, but you must have the VDE CRECTX privilege to create a public context. While a private context can have the same name as a public context, neither two private contexts nor two public contexts can share the same name.

QUALIFIERS

/ARCHITECTURE

Under Construction...

/FACILITY=facil-name
/NOFACILITY

The /FACILITY qualifier specifies the default facility for a context established with the SET CONTEXT command. The default facility is the facility that is assumed when the facility name is omitted from a module name. The /NOFACILITY qualifier specifies that the default facility established by the context is null. If you omit both qualifiers, the current default facility becomes the facility associated with the new context.

/LOG (default)
/NOLOG

Controls whether log messages are printed after a new context is created. The /LOG qualifier causes the messages to be printed and the /NOLOG qualifier suppresses them. The messages indicate that each new context has been created and that the database transaction has successfully committed.

/PRIVATE (default)
/PUBLIC

Specifies whether a private or a public context is created. The /PRIVATE qualifier causes VDE to create a context that is private to you. The /PUBLIC qualifier causes VDE to create a context that is available to all users of the library. You must have the CRECTX privilege to use the /PUBLIC qualifier.

Of /PRIVATE and /PUBLIC, the former is the default.

/REMARK="string"

Specifies a remark string to be stored for the new context. The quoted remark string, which can be up to 132 characters long, may contain any explanatory remark about the new context that you want stored in the VDE database. Typically the remark describes the function or purpose of the context.

/STREAM=stream-name
/NOSTREAM

The /STREAM qualifier specifies that the indicated development stream becomes the default stream when the context is established with the SET CONTEXT command. The /NOSTREAM qualifier specifies that the default stream established by the context is null. If you omit both qualifiers, the current default stream becomes the stream associated with the new context.

EXAMPLES

1

```
VDE> CREATE CONTEXT NEWCTX/STREAM=V2.0/FACILITY=RMS/REMARK="Version 2"
%VDE-I-CTXADDED, context NEWCTX added to the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE> SHOW CONTEXT NEWCTX
Context NEWCTX (private to user SMITH)
   Default stream is V2.0
   Default facility is RMS

VDE>
```

This example creates a private context NEWCTX in which the default stream is V2.0 and the default facility is RMS. The SHOW CONTEXT command displays the context settings associated with the new context.

```
2   VDE> CREATE CONTEXT PUBCTX/PUBLIC/STREAM=V3.3-2FT/FACILITY=RMS
    %VDE-I-CTXADDED, context PUBCTX added to the database
    %VDE-I-COMMIT, database transaction has successfully committed
    VDE>
```

In this example, the CREATE CONTEXT command creates a public context PUBCTX available to all users. When this context is specified on a SET CONTEXT command, the default stream is set to V3.3-2FT and the default facility is set to RMS.

CREATE DIRECTORY_TREE

Creates the disk directory tree for the VDE library's delta files, for the default development stream, or for a specified facility.

Requires CRESTRM privilege.

FORMAT CREATE DIRECTORY_TREE

DESCRIPTION

The CREATE DIRECTORY_TREE command creates the directory tree on disk for a VDE library's delta files, for a development stream, or for a facility. You use the CREATE DIRECTORY_TREE command only if you deferred directory creation by using the /DEFER qualifier on a CREATE LIBRARY, CREATE STREAM, or CREATE FACILITY command. The /DEFER qualifier prevents these three commands from creating the disk directories they normally would create. You then have the opportunity to override the normal directory-creation rules within the SET DIRECTORY commands. After entering your SET DIRECTORY commands, you enter the CREATE DIRECTORY_TREE command to create disk directories according to the rules you specified with your SET DIRECTORY commands.

Directories are created using the protection setting of the parent directory for the UIC group and world access, any applicable ACE entries from the parent directory, and allowing full system and owner access.

QUALIFIERS

/BRIEF (default)
/FULL

Determines how many log messages are displayed when VDE creates new disk directories. The /BRIEF qualifier only displays log messages for the stream or delta-file root directory and for each facility root directory. The /FULL qualifier displays a log message for every directory VDE creates, including all facility subdirectories. (The facility subdirectories are the most numerous directories.) These qualifiers have no effect if you use the /NOLOG qualifier.

Of /BRIEF and /FULL, the former is the default.

/DELTA_FILES
/STAGING
/STREAM (default)

Determines whether directories are created for the VDE library's delta files, for its staging areas, or for the default development stream. The /DELTA_FILES qualifier specifies that disk directories be created for the VDE library's delta files. These are the directories that hold CMS libraries. The /STAGING qualifier specifies that the root directory be created for the library's staging directories. Staging directories are temporary holding areas for files that are replaced into the library

with queued replacements. The /STREAM qualifier specifies that disk directories be created for the default development stream.

If you specify the /DELTA_FILES qualifier but not the /FROM_CMS_ LIBRARY qualifier, VDE creates the CMS library for each facility affected by this command. If you specify the /STREAM qualifier, VDE runs the default stream's new-stream script for each facility affected by this command, provided such a script is defined. Depending on how it is defined, this script may populate the stream's facility-specific directories with source modules and other files.

Of /DELTA_FILES, /STAGING and /STREAM, the latter is the default.

/FACILITY=fac-name

Specifies that disk directories be created only for the facility specified by the **fac-name** parameter. If this qualifier is omitted, directories are created for all facilities, either for delta files or for the default stream.

You can create directory trees for multiple facilities by using wildcard characters in the **fac-name** parameter. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. VDE creates directory trees for those facilities whose names match the wildcard pattern.

/FROM_CMS_LIBRARY[=date-time]

Specifies that the delta-file root directory for each facility covered by the current CREATE DIRECTORY_TREE command already contains a CMS library and that VDE should enter all modules and module generations present in that CMS library into the current VDE database. This qualifier, which VDE also accepts on the CREATE FACILITY command, gives you a way to automatically populate a VDE library with modules from an existing CMS library.

If you specify the optional **date-time** parameter, VDE additionally retrieves all reservations from the CMS library that were created since the date and time given by that parameter. VDE prints warning messages for any reservations created earlier than this time and for any reservations whose stream (CMS class) cannot be ascertained. If you omit the **date-time** parameter, VDE does not retrieve reservations from the CMS library, only modules and generations.

If this qualifier is omitted, VDE creates a new CMS library in each delta-file facility root directory it creates. This qualifier has no effect unless you also use the /DELTA_FILES qualifier.

If this qualifer is specified, all VDE streams *must* be created *prior* to issuing this command, and these stream names must match (a subset of) the CMS class names in use in the CMS libraries; these stream names must match the CMS classes to be uploaded from within the CMS library. In order to correctly establish the main and variant lines of descent and the correct CMS generation number sequence, all streams involved must be created with the appropriate /MAIN, /NOMAIN, /PARENT and /SUCCESSOR settings.

/LOG (default)
/NOLOG

Controls whether log messages are printed when disk directories are created. The /LOG qualifier causes such messages to be printed and the

/NOLOG qualifier suppresses them. These messages indicate the names of the disk directories that VDE creates. The /BRIEF and /FULL qualifiers control how many log messages are printed.

EXAMPLES

1

```
VDE> CREATE LIBRARY/DEFER
Enter the name you want for the new library: LIBNAME
Enter directory specification for the new library: [PROJDIR]
Enter your full name (first and last names): Zrebhasha Fulkutznodoy
Enter the remark for new library: Library for the mumble project
Library LIBNAME to be created in root directory [PROJDIR]
   Library root directory DEV$:[PROJDIR] has been created.
   Logical name VDE$LIBRARY now points to the new library.
   Creating library database; please wait...
   Library database has been created.

Library creation completed.

VDE> SHOW LOCATION
Directory locations for stream MAIN:
   Stream root directory is not recorded in database

VDE> SHOW LOCATION/DELTA_FILES
Directory locations for delta files:
   Delta-file root directory is not recorded in database
```

In this example, the /DEFER qualifier to the CREATE LIBRARY command prevents disk directories from being created for the library's delta files and its default stream MAIN. The CREATE LIBRARY command only creates the library root directory DEV$:[PROJDIR], where it also creates the VDE database. The SHOW LOCATION commands confirm that no other directories are recorded in the VDE database.

2

```
VDE> SET DIRECTORY/DELTA_FILES [PROJDIR]
%VDE-I-DIRADDED, directory specification [PROJDIR] added to the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE> SET DIRECTORY/STREAM DISK2:[MAIN_DIR]
%VDE-I-DIRADDED, directory specification DISK2:[MAIN_DIR] added to the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

A SET DIRECTORY command then specifies that the root directory for the library's delta files (its CMS libraries) should be directory [PROJDIR]. The normal rule used within VDE is to make directory DEV$:[PROJDIR.VDE$CMS] the delta-file root directory and to make subdirectories within that directory contain the CMS libraries for individual facilities. A second SET DIRECTORY command specifies that the root directory for stream MAIN is directory DISK2:[MAIN_DIR]. VDE typically makes the directory DEV$:[PROJDIR.MAIN] the root directory for stream MAIN. In this case, the user wants stream MAIN's whole directory tree on a separate disk.

This sequence sets up the library for the following commands:

3
```
VDE> CREATE DIRECTORY_TREE/DELTA_FILES
Creating directory tree for delta files:
   Root directory WORK4:[PROJDIR] already exists
   Facility CODE root directory DEV$:[PROJDIR.CODE] has been created
      CMS library for facility CODE has been created
   All directories for delta files now exist

%VDE-I-COMMIT, database transaction has successfully committed
VDE>
VDE> CREATE DIRECTORY_TREE/STREAM
Creating directory tree for stream MAIN:
   Root directory DISK2:[MAIN_DIR] has been created
   Facility CODE root directory DISK2:[MAIN_DIR.CODE] has been created
   All directories for stream MAIN now exist

%VDE-I-COMMIT, database transaction has successfully committed
VDE>
VDE> SHOW LOCATION/DELTA_FILES
Directory locations for delta files:
   Delta-file root directory: DEV$:[PROJDIR]

   Directory locations for facility CODE:
      Root directory: DEV$:[PROJDIR.CODE]

VDE> SHOW LOCATION/STREAM
Directory locations for stream MAIN:
   Stream root directory: DISK2:[MAIN_DIR]
      Subdirectory:   DISK2:[MAIN_DIR.VDE$BLD] (symbol VDE$BLD)

   Directory locations for facility CODE:
      Root directory: DISK2:[MAIN_DIR.CODE]
      Subdirectory:   DISK2:[MAIN_DIR.CODE.OBJ] (symbol OBJ)
      Subdirectory:   DISK2:[MAIN_DIR.CODE.SRC] (symbol SRC)
      Subdirectory:   DISK2:[MAIN_DIR.CODE.VDE$COM] (symbol VDE$COM)
      Subdirectory:   DISK2:[MAIN_DIR.CODE.VDE$LOG] (symbol VDE$LOG)

VDE>
```

This section shows the results of two CREATE DIRECTORY_TREE commands. The first one creates the directory tree for the library's delta files and create the CMS library for facility CODE in that directory tree. The second command creates the directory tree for stream MAIN. As the log messages show, the directory tree begins with root directory DISK2:[MAIN_DIR] as requested. The subsequent SHOW LOCATION commands confirm that all disk directories have been created in the desired locations.

4

```
VDE> CREATE FACILITY/DEFER NEWFACIL
Please supply required information for facility NEWFACIL:
Enter a remark for the new facility: New facility
%VDE-I-FACADDED, facility NEWFACIL added to the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
VDE> SET DIRECTORY/FACILITY=NEWFACIL [PROJDIR.NEW]
%VDE-I-DIRADDED, directory specification [PROJDIR.NEW] added to the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
VDE> CREATE DIRECTORY_TREE/FULL/FACILITY=NEWFACIL/DELTA_FILES
Creating directory tree for delta files:
   Facility NEWFACIL root directory DEV$:[PROJDIR.NEWFACIL] has been created
      CMS library for facility NEWFACIL has been created
   All directories for delta files now exist

%VDE-I-COMMIT, database transaction has successfully committed
VDE>
VDE> CREATE DIRECTORY_TREE/FULL/FACILITY=NEWFACIL
Creating directory tree for stream MAIN:
   Facility NEWFACIL root directory DEV$:[PROJDIR.NEW] has been created
      Subdirectory DEV$:[PROJDIR.NEW.OBJ] has been created
      Subdirectory DEV$:[PROJDIR.NEW.SRC] has been created
      Subdirectory DEV$:[PROJDIR.NEW.VDE$COM] has been created
      Subdirectory DEV$:[PROJDIR.NEW.VDE$LOG] has been created
   All directories for stream MAIN now exist

%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

In this example, the CREATE FACILITY command creates a new facility NEWFACIL, but defers the creation of disk directories for that facility. A SET DIRECTORY command then specifies that directory [PROJDIR.NEW] should be the facility root directory for the default stream MAIN. Two CREATE DIRECTORY_TREE commands follow. The first one creates the delta-file root directory and CMS library for the facility. The second one creates the default stream's directory tree for the facility. This directory tree consists of a facility root directory and four facility subdirectories. The /FULL qualifier causes VDE to prints log messages for all directories it creates for the facility; without this qualifier, VDE omits the subdirectory log messages.

5

```
VDE> CREATE STREAM V2.0/DEFER
Please supply required information for stream V2.0:
Enter a remark for the new stream: Version 2.0
%VDE-I-STRMADDED, stream V2.0 added to the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
VDE> SET STREAM V2.0
VDE> SHOW LOCATION/STREAM
Directory locations for stream V2.0:
   Stream root directory is not recorded in database

VDE> SET DIRECTORY/STREAM [PROJDIR.VER_2]
%VDE-I-DIRADDED, directory specification [PROJDIR.VER_2] added to the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
VDE> CREATE DIRECTORY_TREE/STREAM
Creating directory tree for stream V2.0:
   Root directory DEV$:[PROJDIR.VER_2] has been created
   Facility CODE root directory DEV$:[PROJDIR.VER_2.CODE] has been created
   Facility SPEC root directory DEV$:[PROJDIR.VER_2.SPEC] has been created
   Facility TOOL root directory DEV$:[PROJDIR.VER_2.TOOL] has been created
   All directories for stream V2.0 now exist

%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

The CREATE STREAM command creates a new stream called V2.0, but defers creating the disk directory tree for that stream. The SET STREAM command makes V2.0 the default stream, and the SHOW LOCATION command shows that no directories are recorded in the VDE database for that stream. The SET DIRECTORY command specifies that directory [PROJDIR.VER_2] should be the root directory for the stream. The CREATE DIRECTORY_TREE command creates the full directory tree for stream V2.0. This directory tree includes the stream root directory specified by the SET DIRECTORY command, a facility root directory for each facility in the system, and several facility subdirectories for each facility. Because the /FULL qualifier is not specified, VDE only prints log messages for the stream and facility root directories, and not for the facility subdirectories.

CREATE FACILITY

Creates a new facility definition and its associated disk directories in the current VDE library.

Requires CREFAC privilege.

FORMAT

CREATE FACILITY *facil-name [, facil-name...]*

PARAMETER

facil-name

The name of the new facility to be created. This name can be up to 39 characters long and must follow VDE name syntax.

VDE prohibits the creation of any facilities with names that begin with the character sequences ''VDE$'', ''VSC$'', and ''VTSC$'', as these prefixes are used within VDE.

DESCRIPTION

A facility is the highest level of organization in a VDE library and identifies a set of related modules. For example, the RMS facility is composed of all the modules needed to build the Record Management Services portion of the OpenVMS operating system.

All modules within a given facility must have unique names; no two modules in the same facility can have the same module name and module type. If, however, two modules belong to different facilities, they can have the same module name and type.

The VDE library stores modules from separate facilities in separate disk directories. Therefore, you may want to create a separate facility for each component of your software system that is developed independently.

When you create a facility, the CREATE FACILITY command first creates an entry for the new facility in the VDE database. It then creates a disk directory to hold the delta files of the facility's source modules. VDE uses callable CMS to create a CMS library in that directory. When you later create source modules in the facility, VDE uses callable CMS to create delta files for those modules. A delta file (which is identical to a CMS ''element'') contains all past and present versions of your source file in a compressed form.

The CREATE FACILITY command also creates a disk directory structure (directory tree) for the new facility for the default stream. This directory structure is repeated for all streams that are successors of the default stream. These directories will hold all files associated with the facility that are stream-specific. Such files include clear-copy source files, object files, object libraries, executable images, and other derived files. For each stream, this structure consists of the following directories:

- a facility root directory, which is a subdirectory of the stream root directory

- several facility subdirectories that will contain source files, derived files, build log files, and certain other kinds of files. The subdirectories that are created depend on the SET DIRECTORY commands you have used as well as VDE defaults.

If the default disk directory structure is not suitable for the new facility, you can defer its creation until you have defined the structure you want. To defer creation of the directory structure, use the /DEFER qualifier. Then use the SET DIRECTORY command to define the directory structure you want. Finally, use the CREATE DIRECTORY_TREE command to create the desired directory structure.

QUALIFIERS

/DEFER
/NODEFER (default)

Specifies whether the creation of the disk directory structure for the new facility should be postponed. The /DEFER qualifier postpones creation of the disk directory structure and the /NODEFER qualifier causes the directory structure to be created immediately. Use the /DEFER qualifier when the default directory structure is inappropriate for the new facility.

If the /NODEFER qualifier is specified (or /DEFER is omitted), the CREATE FACILITY command first creates the delta-file directory for the facility and creates a CMS library within that directory. It then creates the facility root directory for the default stream. Next it creates the facility subdirectories within that root directory. The subdirectories that are created are determined by the SET DIRECTORY commands entered for the stream and by the default rules of the VDE utility. Once the directory structure exists, source modules and other files can be added to the new facility. The directory-creation sequence is repeated for all successors of the default stream.

If the /DEFER qualifier is specified, the CREATE FACILITY command does not create the delta-file directory, nor the stream-specific root directories, nor the associated subdirectories. You can therefore enter SET DIRECTORY commands to specify the disk directories you want created for the facility. You then use the CREATE DIRECTORY_TREE command to create the directory structure on disk according to the rules specified by the SET DIRECTORY commands and VDE defaults.

/DELETE
/NODELETE

Controls whether the **delete** attribute is set for the new facility. This attribute allows the facility to be deleted with the DELETE FACILITY command. The /DELETE qualifier sets the delete attribute so that the new facility can be deleted. The /NODELETE qualifier clears the delete attribute so that the new facility cannot be deleted unless you first reset this attribute with a MODIFY FACILITY command. Marking a facility as /NODELETE makes it harder to accidentally delete the facility.

If this qualifier is not specified, VDE uses the value of the allow-deletion library attribute. The library allow-deletion default attribute can be set or reset by using the /ALLOW_DELETE qualifier on the CREATE LIBRARY or on the MODIFY LIBRARY command.

/FROM_CMS_LIBRARY[=date-time]

Specifies that the delta-file root directory for the new facility already contains a CMS library and that VDE should enter all modules and module generations present in that CMS library into the current VDE database. This qualifier gives you a way to automatically populate a VDE library with modules from an existing CMS library. If this qualifier is omitted, VDE creates a new CMS library in the facility's delta-file root directory.

If you specify the optional **date-time** parameter, VDE additionally retrieves all reservations from the CMS library that were created since the date and time given by that parameter. VDE prints warning messages for any reservations created earlier than this time and for any reservations whose stream (CMS class) cannot be ascertained. If you omit the **date-time** parameter, VDE does not retrieve reservations from the CMS library, only modules and generations.

This qualifier has no effect if you also specify the /DEFER qualifier. If you defer directory creation, you must use the /FROM_CMS_LIBRARY qualifier on the CREATE DIRECTORY_TREE command that actually creates the delta-file directories for the new facility.

If this qualifer is specified, all VDE streams *must* be created *prior* to issuing this command, and these stream names must match (a subset of) the CMS class names in use in the CMS libraries; these stream names must match the CMS classes to be uploaded from within the CMS library. In order to correctly establish the main and variant lines of descent and the correct CMS generation number sequence, all streams involved must be created with the appropriate /MAIN, /NOMAIN, /PARENT and /SUCCESSOR settings.

/LOG (default)
/NOLOG

Controls whether log messages are printed after the new facilities are created. The /LOG qualifier causes the messages to be printed and the /NOLOG qualifier suppresses them. The messages indicate that each new facility has been created, that its CMS library has been created, that its disk directories have been created, and that the database transaction has successfully committed.

/PROPAGATE (default)
/NOPROPAGATE

Specifies whether automatic change propagation is allowed by default for new modules created in this facility. /PROPAGATE causes new modules added to the facility to be marked as allowing automatic change propagation when reserved and replaced. /NOPROPAGATE marks such modules as not allowing change propagation. When you create new modules, you can override these defaults with qualifiers to the CREATE MODULE command.

/REMARK="string"

Specifies a remark string to be stored for the new facility. The quoted remark string, which can be up to 132 characters long, may contain any explanatory remark about the new facility that you want stored in the VDE database. Typically the remark describes the function or contents of the facility.

Although this qualifier is required, you can specify an empty remark string. If you do not specify the /REMARK qualifier, VDE prompts you for a remark string.

EXAMPLES

1

```
VDE> CREATE FACILITY ACCTNG /REMARK="Accounting function"
Creating directory tree for delta files:
   Facility ACCTNG root directory DEV$:[PROJDIR.VDE$CMS.ACCTNG] has been created
      CMS library for facility ACCTNG has been created
   All directories for delta files now exist

Creating directory tree for stream MAIN:
   Facility ACCTNG root directory DEV$:[PROJDIR.MAIN.ACCTNG] has been created
      Subdirectory DEV$:[PROJDIR.MAIN.ACCTNG.OBJ] has been created
      Subdirectory DEV$:[PROJDIR.MAIN.ACCTNG.SRC] has been created
      Subdirectory DEV$:[PROJDIR.MAIN.ACCTNG.VDE$COM] has been created
      Subdirectory DEV$:[PROJDIR.MAIN.ACCTNG.VDE$LOG] has been created
   All directories for stream MAIN now exist

%VDE-I-FACADDED, facility ACCTNG added to the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example creates a new facility called ACCTNG. The remark string indicates that this facility performs an accounting function in the user's software system. The log messages indicate that the delta-file disk directory has been created, that a CMS library has been established in that directory, that a facility root directory and various facility subdirectories have been created for stream MAIN, and that the facility was successfully added to the database.

CREATE FOLD

Creates a new fold record in the VDE database. A fold record is a reminder that a source change should be manually "folded" into a given development stream in order to propagate the change.

Requires RESREP privilege.

FORMAT

CREATE FOLD *mod-name [, mod-name...]*

PARAMETER

mod-name

Specifies a module for which a fold record is to be created. The module name consists of an optional facility name enclosed in square brackets, a module name, and an optional type name preceded by a period (such as [FACIL]MOD1.MAR). If the facility name is omitted, the module is assumed to belong to the default facility. If the type name is omitted, VDE creates fold records for all source modules with the specified module name in the given facility.

You can create fold records for more than one module by using wildcard characters in any of the three components of the module name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. VDE creates fold records for the source modules whose names match the wildcard pattern.

DESCRIPTION

A fold record is a record in the VDE database that reminds a user to manually ''fold'' a source change from one development stream into another. A fold record is created when the REPLACE or PERFORM REPLACEMENT command attempts to propagate a source change from a given development stream to one of its successors, but finds that it cannot automatically propagate the change because the source module has already diverged between the two streams. The user must then reserve the module in the successor stream, manually edit the change into that generation of the module, and then replace it. This operation is known as ''folding'' the change into the successor stream. Once the fold operation has been performed, its fold record should be deleted from the VDE database with the /FOLD qualifier to the REPLACE command or with the CANCEL FOLD command.

The CREATE FOLD command lets you manually enter a fold record into the VDE database. This fold record and its remark string can serve as a reminder to perform some action on a specified module in a specified stream. Once you have performed this action, you can delete the fold record with the CANCEL FOLD command. The new fold record records that some change should be propagated from the latest generation of the specified module in the default development stream into another development stream that you specify.

QUALIFIERS

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want a fold record created for each module. The /CONFIRM qualifier causes VDE to print a message for each module asking whether you want to create a fold record for that module. If you answer YES (or Y), VDE creates a fold record for it. If you answer NO (or N), VDE does not create a fold record for it. If you enter a module name with a wildcard specification, VDE asks you to confirm the creation of a fold record for each module whose name matches the wildcard specification. The /NOCONFIRM qualifier causes VDE to create the fold records without asking for confirmation.

/FROM_STREAM=stream-name

Specifies the development stream from which the source change should be folded. The **stream-name** parameter gives the name of the stream. If you omit this qualifier, VDE determines what generation you want to fold by looking up the latest generation for the current default stream. VDE then records that you want to fold the change from the stream in which that generation was created.

/IDENTIFICATION=fold-ident

Specifies a fold identifier for the new fold record. Each fold record has a unique fold identifier so that you can distinguish between multiple fold records for the same module and user in the same stream. By default, the CREATE FOLD command assigns a unique numeric identifier to each fold record, such as ''1'', ''2'', and so on. The /IDENTIFICATION qualifier overrides this default and allows you to select a fold identifier of your choice, such as a mnemonic name. The **fold-ident** parameter specifies the fold identifier. It can be up to 39 characters long and must follow VDE name syntax.

/LOG (default)
/NOLOG

Controls whether log messages are printed when the new fold records are created. The /LOG qualifier causes the messages to be printed and the /NOLOG qualifier suppresses them. The messages indicate that a fold record has been created for each specified module and that the database transaction has successfully committed.

/REMARK="string"

Specifies a remark string to be stored for each new fold record. The quoted remark string, which can be up to 132 characters long, may contain any explanatory remark about the fold to be performed that you want stored in the VDE database. Typically the remark describes the function or contents of the source change to be folded into the new stream.

Although this qualifier is required, you can specify an empty remark string. If you do not specify the /REMARK qualifier, VDE prompts you for a remark string.

/STREAM=stream-name

Specifies the development stream into which the source change should be folded. The **stream-name** parameter gives the name of the stream. This qualifier is required; if you do not specify it, VDE prompts you for the stream name.

/USERNAME=username

Specifies that the fold record be created on behalf of another user. The **username** parameter specifies the OpenVMS username of the other user. You must have the USERNAME privilege to use this qualifier.

EXAMPLES

1

```
VDE> CREATE FOLD X.FOR/STREAM=V1.0/REMARK="Fix access violation in sort routine"
%VDE-I-FOLDADDED, fold record 1 for module [CODE]X.FOR added to database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
VDE> SHOW FOLD
Fold 1 of [CODE]X.FOR;2(2) into stream V1.0           26-JUN-1990 JONES
VDE>
```

This example creates a new fold record for module [CODE]X.FOR. The remark indicates that a sort routine access violation should be fixed in that module. The change should be made in stream V1.0. The log message shows that VDE assigned the unique fold identifier 1 to the new fold record. The SHOW FOLD command shows that the fold record is now in the VDE database.

CREATE GROUP

Creates a new group definition in the VDE database. A group is either a source group or a build group. A source group consists of a set of source modules to be acted upon when the group name is specified with various source control commands. A build group consists of a set of modules to be built when the group name is specified with a BUILD command.

Requires CREGRP privilege.

FORMAT

CREATE GROUP *group-name = mod-name [, mod-name...]*

PARAMETERS

group-name

The name of the new group to be created. This name may be up to 39 characters long and must follow VDE name syntax.

mod-name

The name of a module to be included in the group. The module name consists of an optional facility name enclosed in square brackets, a module name, and an optional type name preceded by a period. An example is [FACIL]MOD1.PAS. If no facility name is specified, the current default facility is assumed. If no type name is specified, all source modules (for a source group) or all module (for a build group) with the given module name in the given facility are included in the group.

You can add many modules to the group with one **mod-name** parameter by specifying wildcard characters in any of the three components of the module name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Those source modules (for a source group) or modules (for a build group) whose names match the wildcard pattern become members of the new group.

DESCRIPTION

A group consists of the set of modules that are members of the group. A source group only consists of source modules. When the name of a source group is specified with the RESERVE, REPLACE, FETCH, UNRESERVE, or SHOW RESERVATION command, that command acts on all modules that are members of the group. A build group can consists of both source modules and derived modules, although it normally only consists of executable images and other derived modules that can be the outputs of a system build. When the name of a build group is specified with the BUILD command, that command builds all modules that are members of the group.

When a new group is created, it is only defined in the default development stream. You can refer to the new group only from within the stream to which it belongs.

CREATE GROUP

When you want to create the same group for more than one development stream, you must use the CREATE GROUP command for each stream. For each stream, first enter the SET STREAM command to set the development stream, then enter the CREATE GROUP command to create the desired group within the stream.

When a new development stream is created, all group definitions from the parent stream are copied to the child stream. If modified, the group definitions then evolve independently in the two development streams.

QUALIFIERS

/BUILD
/SOURCE (default)

Specifies whether a source group or build group is created. The /SOURCE qualifier causes VDE to create a source group and the /BUILD qualifier causes VDE to create a build group. If neither qualifier is specified, VDE creates a source group.

Of /BUILD and /SOURCE, /SOURCE is the default.

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each module matching the **mod-name** parameters added to the group. The /CONFIRM qualifier causes VDE to print a message for each matching module asking whether you want that module added. If you answer YES (or Y), the module is added to the group. If you answer NO (or N), the module is not added to the group. The /NOCONFIRM qualifier causes VDE to add all modules matching the **mod-name** parameters to the group without asking for confirmation.

/LOG (default)
/NOLOG

Controls whether log messages are printed after the new group has been created. The /LOG qualifier causes the messages to be printed and the /NOLOG qualifier suppresses them. The messages indicate that the new group has been created, that the requested modules are members of the group, and that the database transaction has committed successfully.

/REMARK="string"

Specifies a remark string to be stored for the new group. The quoted remark string, which can be up to 132 characters long, may contain any explanatory remark about the new group that you want stored in the VDE database. Typically the remark describes the purpose or contents of the group.

EXAMPLES

1

```
VDE> CREATE GROUP/REMARK="Two modules" GRP1 = A.PAS, B.MAR
%VDE-I-GRPADDED, group GRP1 added to the database
%VDE-I-GRPADD, module [CODE]A.PAS is added to group GRP1
%VDE-I-GRPADD, module [CODE]B.MAR is added to group GRP1
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This command creates a new source group called GRP1. The new group contains two members, source modules A.PAS and B.MAR. These modules are reserved if the group name is specified with the RESERVE command, for example. Both modules are assumed to belong to the current default facility, facility CODE in this example. The remark string ''Two modules'' is stored in the database with the rest of the group definition and can be retrieved later with the SHOW GROUP command. The example shows the informational messages created by the /LOG qualifier, specified by default in this example.

2

```
VDE> CREATE GROUP/BUILD SYSGRP = X.EXE, Y.EXE, Z.HLB
%VDE-I-GRPADDED, group SYSGRP added to the database
%VDE-I-GRPADD, module [BUILD_TEST]X.EXE is added to group SYSGRP
%VDE-I-GRPADD, module [BUILD_TEST]Y.EXE is added to group SYSGRP
%VDE-I-GRPADD, module [BUILD_TEST]Z.HLB is added to group SYSGRP
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This command creates a new build group called SYSGRP. The build group contains three members, executable images X.EXE and Y.EXE and the help library Z.HLB. If group name SYSGRP is specified as a parameter to a BUILD command, VDE will build these three modules.

CREATE LIBRARY

Creates a new VDE library, including its database and disk directory structure.

FORMAT

CREATE LIBRARY *dir-spec*

PARAMETER

dir-spec

Specifies the name of the root directory of the library. The root directory is the disk directory that contains the database of the library. It normally also contains subdirectories that hold the contents of the library. Use the OpenVMS directory specification format to specify the **dir-spec** parameter.

This parameter is required. If not specified, VDE prompts you for the name of the root directory.

DESCRIPTION

A VDE library is the repository for all the modules in your software system. A library consists of a top-level disk directory called the root directory, an Rdb database which resides in the root directory, and a number of directories (or subdirectories) which contain all the source modules, derived modules, and other components of your software system.

The CREATE LIBRARY command first creates a root directory for the new library and defines logical name VDE$LIBRARY to contain the name of this directory. It then creates and initializes a VDE database in the root directory. The CREATE LIBRARY command normally also creates the disk directory structure that will hold all sources and other files that make up your software system. However, if the default directory structure is not appropriate for you, you can use the /DEFER qualifier to postpone directory creation until you have specified the directory structure that you want.

The VDE database is a multifile Rdb database. This database consists of three separate files: an Rdb root file (the .RDB file), a storage area file (the .RDA file), and a snapshot file (the .SNP file). The .RDB file is always stored in the VDE library's root directory. This file is relatively small. The .RDA file stores the bulk of the database contents and the .SNP file allows multiple users to use the database concurrently. The .RDA and .SNP files are stored in the VDE library's root directory by default, but you can specify different directory locations for these files with the /DATABASE and /SNAPSHOT qualifiers, or by defining the logical names VDE$DB_ AREA and VDE$SNAPSHOT. By placing these files on different disks, you can distribute your VDE library's space and I/O requirements across multiple disks.

You may also want to establish an Rdb after-image journal file for the VDE database. If used with the RMU database backup facility, such a journal file allows Rdb to recover all database data as of the last completed transaction after a disk failure. To establish journalling, use the /JOURNAL qualifier to specify the directory location of the journal file.

This directory should always be on a different physical disk than the rest of your VDE database.

Once you have created a new VDE library, that library becomes the current VDE library for your present process so that subsequent VDE commands act upon that library. To change the current VDE library, you must use the SET LIBRARY command or redefine logical name VDE$LIBRARY to point to the root directory of the VDE library you want to work with.

Most steps in creating a new VDE library are reasonably fast. For example, each directory in your directory structure is created in a fraction of a second unless your system is heavily loaded. However, VDE requires several minutes to create and initialize the library database. The database can usually be created in about six to eight minutes but more time may be required on a heavily loaded system. A log message tells you when database creation starts.

QUALIFIERS

/ALLOW_DELETE (default)
/NOALLOW_DELETE

Specifies the default value for the **delete** attribute for new modules, facilities, and streams. The /ALLOW_DELETE qualifier causes VDE to set the **delete** attribute by default when users create new such entities. As a result, users can delete these entities without first having to set the **delete** attribute with a separate MODIFY command. The /NOALLOW_ DELETE qualifier causes VDE to clear the **delete** attribute by default for new modules, facilities, and streams. As a result, it is harder for a user to delete an entity by mistake; the entity's **delete** attribute must be set with a separate MODIFY command before the entity can be deleted with a DELETE command. When creating new modules, facilities, and streams, you can override this default setting with an explicit /DELETE or /NODELETE qualifier to the CREATE command.

Use the /ALLOW_DELETE qualifier if you expect to make frequent changes to your library, often deleting modules, facilities, and streams. Use the /NOALLOW_DELETE qualifier if you are more concerned about preventing accidental deletion of such entities.

/ASK_INFO_FILE
/NOASK_INFO_FILE (default)

Specifies whether the REPLACE command should by default ask the user whether he or she wants to edit an information file when creating a queued replacement. The /ASK_INFO_FILE qualifier sets a library attribute that causes the REPLACE command to ask for an information file by default. The /NOASK_INFO_FILE qualifier clears this library attribute so that the REPLACE command does not ask for an information file by default. This library attribute only affects queued replacements (not immediate replacements), and it only affects the REPLACE command's default behavior; users can always use the command's /INFORMATION or /NOINFORMATION qualifier to explicitly request either behavior.

/ASK_REPL_CAUSE
/NOASK_REPL_CAUSE (default)

Specifies whether the REPLACE command should by default ask the user for the ‘‘cause’’ of the replacement. The cause may be a fold operation, a Source Control Tracking (SCT) entry in a Notes conference, or some other cause. The /ASK_REPL_CAUSE qualifier sets a library attribute that causes the REPLACE command to query the user for the cause of the replacement. The /NOASK_REPL_CAUSE qualifier clears this library attribute so that the REPLACE command does not ask for this information.

/AUTH_PRIVILEGES=(priv [,priv...])

Specifies the standard authorized privileges for the library. These are the authorized privileges that a new user gets when added to the library unless a different set of authorized privileges is explicitly specified for that user. Authorized privileges are those VDE privileges the user is allowed to set with the SET PRIVILEGE command.

If only one privilege is specified, the parentheses may be omitted. If this qualifier is omitted entirely, VDE provides default values for the standard authorized privileges of the library.

Each **priv** parameter specifies the name of one privilege. For a list of privileges, see Table 3–5 in SET PRIVILEGES.

/AUTO_ADD_USER[=rights-ident]
/NOAUTO_ADD_USER (default)

Specifies whether new users should be added to the library database automatically when they first access the library. The /AUTO_ADD_USER qualifier specifies that new users should be added to the library database automatically when they first access the library. If the **rights-ident** parameter is specified, new users are only added to the library if they hold the OpenVMS rights identifier specified by the parameter. The /NOAUTO_ADD_USER qualifier specifies that new users not be added to the library database automatically. In this case, new users can only be added with explicit CREATE USER commands.

The owner information specified in the OpenVMS SYSUAF database will be used as the owner of any new VDE username created. VDE will ignore any leading numerics, leading hyphens, leading space characters, and leading tilde characters, listed in the SYSUAF owner field.

/AUTO_CONVERT
/NOAUTO_CONVERT (default)

Specifies whether the library and its database are automatically converted to the new library format when a new version of VDE is used on the library. A new version of VDE will frequently require a different database schema and possibly other changes to a library than the previous version. Each new version of VDE can convert a VDE library from an older format to the current format. This is either done automatically the first time someone accesses the library database, or it is done with an explicity CONVERT LIBRARY command. These qualifiers determine the behaviour.

The /AUTO_CONVERT qualifier specifies that the library should be converted automatically the first time a user uses a new version of VDE on the present library and its database. /NOAUTO_CONVERT specifies that the library should not be converted automatically; instead a privileged user must explicitly convert the library with the CONVERT LIBRARY command.

The advantage of automatic conversion is that it is automatic and requires no special steps on the part of the user. However, once a library has been converted, it is not possible to use an older version of VDE on that library.

The advantage of disabling automatic conversion is that you get a chance to back up the database (or the whole library) before you explicitly convert the library. You then have the option of restoring the backed up copy of the library if you need to go back to using the old version of VDE. However, once you install a new version of VDE, nobody can use the existing library until it is converted. Also, all accessors must use the same or a compatible version of VDE.

In general, automatic conversion is best for small VDE libraries whose databases are relatively easy to rebuild if necessary. Explicit conversion may be more suitable for large libraries where the difficulty of rebuilding the library justifies more stringent backup and recovery procedures.

/NOAUTO_CONVERT is the default.

/AUTO_MAIL (default)
/NOAUTO_MAIL

Specifies whether VDE should automatically send mail notification to the user who has queued a replacement when that replacement is later performed. The /AUTO_MAIL qualifier sets a library attribute that causes the PERFORM REPLACEMENT command to send such mail notification. The /NOAUTO_MAIL qualifier clears this library attribute so that the PERFORM REPLACEMENT command does not send such notification.

/CMS_ELEM_HIST (default)
/NOCMS_ELEM_HIST

Specifies whether VDE creates the CMS elements for new modules with default CMS history and notes strings. The /CMS_ELEM_HIST qualifier causes VDE to create the CMS elements for new modules with default history and notes strings and the /NOCMS_ELEM_HIST qualifier causes VDE to create CMS elements with no default history or notes strings. /NOCMS_ELEM_HIST is normally only used if you regularly fetch VDE modules using the CMS FETCH command and you want to make sure you do not get CMS history or notes by default. If you always fetch through VDE, use /CMS_ELEM_HIST because it allows VDE to avoid a deficiency in CMS and always give and remove history information correctly.

/CREATOR_NAME="string"

Specifies the full name of the user who created the library. The quoted name string, which may be up to 40 characters long, should contain your first name, middle initial, and last name. The CREATE LIBRARY command automatically creates a user record in the new library for the library's creator. This name string becomes part of that user record.

This qualifier is required. An empty name string may be specified but this is not recommended. If the /CREATOR_NAME qualifier for a library is not specified, VDE prompts you for the name string.

/DATABASE=dir-spec

Specifies that the database storage area (.RDA) file and the snapshot (.SNP) file be created in the directory location given by the **dir-spec** parameter. This qualifier lets you put the largest part of the VDE database on a different disk than the rest of the VDE library. If you omit this qualifier, VDE creates the .RDA and .SNP files in the VDE library's root directory.

/DEFER
/NODEFER (default)

Specifies whether the creation of the directory structure for the new library should be postponed. The /DEFER qualifier postpones creation of the disk directory structure for the library and the /NODEFER qualifier causes the directory structure to be created immediately. Use the /DEFER qualifier when the default directory structure is inappropriate for the new library.

If the /NODEFER qualifier is specified (or /DEFER is omitted), the CREATE LIBRARY command first creates the root directory and the VDE database for the library. Then it creates the remaining disk directory structure for the library. This structure, which includes directories to hold delta-files for one facility and directories to hold files for one development stream, is determined by the default rules of the utility. Once the directory structure exists, source modules and other files can be added to the new library.

If the /DEFER qualifier is specified, the CREATE LIBRARY command creates only the library's root directory and its database; it does not create the remaining directory structure for the new library. If you use the /DEFER qualifier, you can enter SET DIRECTORY commands to specify the disk directories you want for the library. Then use the CREATE DIRECTORY_TREE command to create that directory structure on disk. The CREATE DIRECTORY_TREE command uses VDE defaults combined with the rules you specified with SET DIRECTORY commands to create the new directory structure.

/DEF_PRIVILEGES=(priv [,priv...])

Specifies the standard default privileges for the library. These are the default privileges that a new user gets when added to the library unless a different set of default privileges is explicitly specified for that user. Default privileges are those VDE privileges that are enabled each time the user starts a VDE session.

If only one privilege is specified, the parentheses may be omitted. If this qualifier is omitted entirely, VDE provides default values for the standard default privileges of the library.

Each **priv** parameter specifies the name of one privilege. For a list of privileges, see Table 3–5 in SET PRIVILEGES.

Each privilege specified becomes a standard authorized privilege as well as a standard default privilege. This rule ensures that users are authorized to have all their default privileges.

/HISTORY_NOTES (default)
/NOHISTORY_NOTES

Specifies whether VDE's RESERVE command can produce CMS history or notes information in its output files. The /HISTORY_NOTES qualifier allows the RESERVE command to produce history and notes information

and to accept the /HISTORY, /NOTES, and /POSITION qualifiers. The /NOHISTORY_NOTES qualifier causes the RESERVE command to never produce history and notes information (even when such information is the default for a module being reserved) and to not accept the /HISTORY, /NOTES, and /POSITION qualifiers. In this case, you must use the VDE FETCH command to get CMS history or notes information for a module. Disabling history and notes information in reserved files makes certain mistakes less likely where such information gets replaced into the VDE library as part of the module text.

/JOURNAL=dir-spec

Enables after-image journalling and specifies that a database after-image journal (.AIJ) file be created in the directory location given by the **dir-spec** parameter. When used with the RMU backup and restore facilities, after-image journalling allows you to recover the up-to-date contents of your database after a disk failure. The specified directory should always reside on a different physical disk than any of your other database files so that the journal file cannot be damaged by the same disk failure as the database itself. If you omit this qualifier, VDE disables after-image journalling. Digital recommends that you always use journalling.

/LOG (default)
/NOLOG

Controls whether log messages are printed when the new library is created. The /LOG qualifier causes messages to be printed and the /NOLOG qualifier suppresses them. The messages indicate that the library and its many components have been created.

/NAME=lib-name

Specifies a name for the new library to be created. This name may be up to 39 characters long and must follow VDE name syntax. VDE stores this name in the database and displays it back to you when you enter the SHOW LIBRARY or SHOW DEFAULTS command. This qualifier is required; if you do not specify it, VDE prompts you for the library name.

/REMARK="string"

Specifies a remark string. The quoted remark string, which can be up to 132 characters long, may contain any explanatory remark about the new library that you want stored in the library's database. Typically the remark describes the purpose or contents of the new library.

Although this qualifier is required, you can specify an empty remark string. If you do not specify the /REMARK qualifier, VDE prompts you for a remark string.

/REPL_HISTORY (default)
/NOREPL_HISTORY

Specifies whether VDE should maintain a history of all past source code replacements into the VDE library. The /REPL_HISTORY qualifier specifies that such history information be maintained in the VDE database. The /NOREPL_HISTORY qualifier specifies that such information not be maintained. When you maintain replacement history, you can display past replacements using the /HISTORY qualifier to the SHOW REPLACEMENT command.

For documentation on the SHOW REPLACEMENT command in SHOW REPLACEMENT.

/SEND_MAIL (default)
/NOSEND_MAIL

Specifies whether VDE should send mail notification messages for those VDE operations that normally send such messages. The /SEND_MAIL qualifier enables VDE to send such mail messages and the /NOSEND_MAIL qualifier suppresses all VDE mail messages. The /NOSEND_MAIL qualifier is useful when you want to test or play with a private VDE library without sending notification messages to other people. However, for normal production use, you should always leave the mail notification feature enabled. The CREATE LIBRARY command enables mail notification by default.

/SHOW_ARCH_NAME
/NOSHOW_ARCH_NAME (default)

Specifies whether the architecture name is displayed as part of every module name in VDE output. The /SHOW_ARCH_NAME qualifier sets a library attribute that causes VDE to always include the architecture name in every module name it displays. This attribute also causes the SHOW DEFAULTS command to display the default architecture. The /NOSHOW_ARCH_NAME qualifier clears this library attribute so that VDE omits the architecture names when displaying module names. You normally only want to see the architecture name if you have defined multiple architectures for your VDE library.

This qualifier is not yet implemented.

/SHOW_GEN_EXPR
/NOSHOW_GEN_EXPR (default)

Specifies whether the expected replace-time CMS generation expression for each module is displayed by the RESERVE command. The /SHOW_GEN_EXPR qualifier sets a library attribute that causes the RESERVE command to compute and display the CMS generation expression that the corresponding REPLACE command is most likely to use for the new module generation. You can use this value as an ‘‘audit trail identifier’’ in your code if your project uses this convention. This library attribute also causes the SHOW RESERVATION command to display the expected CMS generation expression and it causes the REPLACE or PERFORM REPLACEMENT command to print an informational message if the actual CMS generation expression for the replaced generation turns out to be different than the expected one. The /NOSHOW_GEN_EXPR qualifier disables this library attribute.

Please note that VDE cannot guarantee that the expected CMS generation expression will actually be used. It is not possible to determine at RESERVE time whether a CMS variant letter should be used when the module is eventually replaced; that decision can only be made at REPLACE time and depends on propagation decisions and stream successor relationships at that time. However, the expected CMS generation expression is generally the one that is actually used.

/SNAPSHOT=dir-spec

Specifies that the database snapshot (.SNP) file be created in the directory location given by the **dir-spec** parameter. This qualifier lets you put the snapshot file on a separate disk to better use disk space or to improve database performance. If you omit this qualifier, VDE creates the .SNP file in the directory given by the /DATABASE qualifier if that qualifier is

specified. If neither qualifier is specified, VDE creates the snapshot file in the VDE library's root directory.

/LOCK
/UNLOCK

The /LOCK and /UNLOCK qualifiers provide a method by which RESERVE, REPLACE, UNRESERVE, and other similar commands can be disabled on a copy of a VDE library. This mechanism is not intended to completely lock-out all access to a VDE library.

Specification of /LOCK write-locks the library, and the corresponding /UNLOCK qualifier reverses the lock. The write-lock status is read only when the user initially accesses the library—once a user has connected to the library, the lock status is *not* updated.

A write-locked library can be used to prevent various user modifications to the contents of the library during library maintenance, or to maintain a shadow copy of a library.

Users holding the MODLIB privilege are allowed to override the library write-lock. An informational message will be displayed when the lock is overridden.

/STATISTICS

Enables statistics collection on the specified library. This will cause various extra questions to be asked during replacement and related operations. This includes the addition of questions around the reason for the change, the number of defects fixed, and the particular project a change is associated with.

/WILD_INS_GEN
/NOWILD_INS_GEN (default)

Specifies whether wildcard characters are allowed in the parameters to the INSERT GENERATION command when the /GENERATION qualifier is specified. The /WILD_INS_GEN qualifier causes VDE to allow wildcard characters in such parameters while the /NOWILD_INS_GEN qualifier causes VDE to not allow such wildcard characters. When the /GENERATION qualifier to INSERT GENERATION specifies a specific CMS generation expression, it is unlikely that this generation expression could apply to more than one module. As a result, it is unlikely that the use of wildcard characters in the module-name parameters will give the result the user intended. To prevent mistakes, VDE therefore does not allow such wildcard usage by default. However, there are some situations when such usage might be correct (for example, if the specified CMS generation expression is 1). For such situations, you can enable the use of wildcard characters with the /WILD_INS_GEN qualifier.

EXAMPLES

1

```
VDE> CREATE LIBRARY [PROJDIR] /NAME=NEWLIB -
_VDE>                         /CREATOR_NAME="John Doe" -
_VDE>                         /REMARK="Main project library"
Library NEWLIB to be created in root directory [PROJDIR]
   Library root directory DEV$:[PROJDIR] already exists.
   Logical name VDE$LIBRARY now points to the new library.
   Creating library database; please wait...
   Library database has been created.

Staging area root directory DEV$:[PROJDIR.VDE$STAGE] has been created

Creating directory tree for delta files:
   Root directory DEV$:[PROJDIR.VDE$CMS] has been created
   Facility CODE root directory DEV$:[PROJDIR.VDE$CMS.CODE] has been created
      CMS library for facility CODE has been created
   All directories for delta files now exist

Creating directory tree for stream MAIN:
   Root directory DEV$:[PROJDIR.MAIN] has been created
      Subdirectory DEV$:[PROJDIR.MAIN.VDE$BLD] has been created
   Facility CODE root directory DEV$:[PROJDIR.MAIN.CODE] has been created
      Subdirectory DEV$:[PROJDIR.MAIN.CODE.SRC] has been created
      Subdirectory DEV$:[PROJDIR.MAIN.CODE.OBJ] has been created
      Subdirectory DEV$:[PROJDIR.MAIN.CODE.VDE$COM] has been created
      Subdirectory DEV$:[PROJDIR.MAIN.CODE.VDE$LOG] has been created
   All directories for stream MAIN now exist

Library creation completed.

VDE>
```

This command creates a new library called NEWLIB. The root directory for the new library is DEV$:[PROJDIR] and the name of the creator is John Doe. The log messages indicate when the various steps are completed. First VDE indicates the root directory of the library currently exists and binds the logical name VDE$LIBRARY to that directory. Then VDE creates the database of the library, which may take several minutes. Next, VDE creates the directory structure for the delta files of the project. Finally, it creates the directory structure for stream MAIN, the only stream in the library.

2

```
VDE> CREATE LIBRARY
Enter directory specification for the new library: [PROJDIR]
Enter the name you want for the new library: NEWLIB
Enter your full name (first and last names): John Doe
Enter the remark for new library: Main project library
Library NEWLIB to be created in root directory [PROJDIR]
   Library root directory DEV$:[PROJDIR] already exists.
   Logical name VDE$LIBRARY now points to the new library.
   Creating library database; please wait...
   Library database has been created.

Creating directory tree for delta files:
   Root directory DEV$:[PROJDIR.VDE$CMS] has been created
   Facility CODE root directory DEV$:[PROJDIR.VDE$CMS.CODE] has been created
      CMS library for facility CODE has been created
   All directories for delta files now exist

Creating directory tree for stream MAIN:
   Root directory DEV$:[PROJDIR.MAIN] has been created
   Facility CODE root directory DEV$:[PROJDIR.MAIN.CODE] has been created
      Subdirectory DEV$:[PROJDIR.MAIN.CODE.SRC] has been created
      Subdirectory DEV$:[PROJDIR.MAIN.CODE.OBJ] has been created
      Subdirectory DEV$:[PROJDIR.MAIN.CODE.VDE$COM] has been created
      Subdirectory DEV$:[PROJDIR.MAIN.CODE.VDE$LOG] has been created
   All directories for stream MAIN now exist

Library creation completed.
```

```
VDE>
```

In this example, the user enters the keywords CREATE LIBRARY and VDE prompts for all other needed information. The new library is created exactly as in the previous example.

```
$
$! Locate and delete previous incarnations of the test facility files,
$! then unpack the saveset containing the test files...
$
$ if f$search("sys$scratch:A.REQ") .nes. "" Then Delete sys$scratch:A.REQ;*/lo
$ if f$search("sys$scratch:B.REQ") .nes. "" Then Delete sys$scratch:B.REQ;*/lo
$ if f$search("sys$scratch:C.B32") .nes. "" Then Delete sys$scratch:C.B32;*/lo
$ if f$search("sys$scratch:D.B32") .nes. "" Then Delete sys$scratch:D.B32;*/lo
$ if f$search("sys$scratch:E.B32") .nes. "" Then Delete sys$scratch:E.B32;*/lo
$ if f$search("sys$scratch:F.B32") .nes. "" Then Delete sys$scratch:F.B32;*/lo
$
$ Backup VDE$CREATE.BCK/SAVE Sys$Scratch:
$
$! Create the core directory -- this must be present...
$
$ Create/Directory STAR$DATA:[QTV.WWW.VDE$DATABASE]
$ VDE
!
! create the WWW library in STAR$DATA:[QTV.WWW...], using the
! mnemonic WWW, placing all Rdb-related library components in
! STAR$DATA:[QTV.WWW.VDE$DATABASE].
!
create librar  STAR$DATA:[QTV.WWW.VDE$DATABASE] /DEFER -
 /NAME=WWW -
 /REMARK="QTV WWW Source Library" -
 /NOAUTO_CONVERT /AUTO_ADD_USER=VMS_SOURCE /AUTO_MAIL -
 /ASK_INFO_FILE /ASK_REPL_CAUSE /SHOW_GEN_EXPR /NOSTATISTI -
 /NOHISTORY_NOTES /NOALLOW_DELETE /NOWILD_INS_GEN -
 /NOCMS_ELEM_HIST /CREATOR="Stephen Hoffman" -
 /DEF_PRIVILEGES=(CREMOD,RESREP,CREFAC) -
 /AUTH_PRIVILEGES=(CREMOD,RESREP,CREFAC) -
 /data=STAR$DATA:[QTV.WWW.VDE$DATABASE] -
 /journ=STAR$DATA:[QTV.WWW.VDE$DATABASE] -
 /snap=STAR$DATA:[QTV.WWW.VDE$DATABASE]
!
SHOW DEFAULTS
SHOW LIBRARY/FULL
MODIFY CONTEXT/PUBLIC DEFAULT/NOSTREAM/NOFACILITY
!
CREATE USER SYSTEM /NAME="System Manager" -
    /DEF_P=ALL/AUTH_P=ALL/REMARK="System Manager"
!
! Create the CMS ("delta") library directory structure.
! Configure the CMS libraries to use the directory structure:
!   STAR$DATA:[QTV.WWW.CMS$DATABASE.'facility'.CMS].
! Configure the "marker" files to use the directory structure:
!   STAR$DATA:[QTV.WWW.CMS$DATABASE.'facility'.SRC]...
!
SET DIRECTORY/DELTA_FILES STAR$DATA:[QTV.WWW.CMS$DATABASE]
SET DIRECTORY/DELTA_FILES/SUBDIRECTORY=VDE$CMS [.CMS]
SET DIRECTORY/DELTA_FILES/SUBDIRECTORY=VDE$MARKER [.SRC]
CREATE DIRECTORY_TREE/DELTA_FILES
SHOW DIRECTORY/DELTA_FILES
!
! Create the staging area...
!
SET DIRECTORY/STAGING STAR$DATA:[QTV.WWW.VDE$STAGING]
CREATE DIRECTORY_TREE/STAGING
!
! Create the stream/build directories
!
SET DIRECTORY/STREAM STAR$DATA:[QTV.WWW.VDE$STREAM]
CREATE DIRECTORY_TREE/STREAM
```

```
!
! create and load the VDE test facility.
!
CREATE FACILITY AAA_VDE_TEST -
    /REMARK="Test facility for VDE developers"
SET FACILITY AAA_VDE_TEST"
CREATE MODULE/NODELETE/REMARK="VDE self-test module" -
  /INPUT=SYS$SCRATCH: A.REQ
CREATE MODULE/NODELETE/REMARK="VDE self-test module" -
  /INPUT=SYS$SCRATCH: B.REQ
CREATE MODULE/NODELETE/REMARK="VDE self-test module" -
  /INPUT=SYS$SCRATCH: C.B32
CREATE MODULE/NODELETE/REMARK="VDE self-test module" -
  /INPUT=SYS$SCRATCH: D.B32
CREATE MODULE/NODELETE/REMARK="VDE self-test module" -
  /INPUT=SYS$SCRATCH: E.B32
CREATE MODULE/NODELETE/REMARK="VDE self-test module" -
  /INPUT=SYS$SCRATCH: F.B32
!
EXIT
$
$ Exit
```

CREATE MODULE

Creates a new module in the current VDE library.

Requires CREMOD privilege.

FORMAT

CREATE MODULE *mod-name [, mod-name...]*

PARAMETER

mod-name

The name of the module being created. The module name must consist of a module name and a module type separated by a period (as in MODNAM.FOR). It may optionally be preceded by a facility name enclosed in square brackets (as in [ACCTNG]MODNAM.FOR where ACCTNG is the facility name). If no facility name is specified, the module is assumed to belong to the current default facility. The facility name, module name, and type name may each be up to 39 characters long and must follow VDE name syntax.

DESCRIPTION

The CREATE MODULE command creates a new module in the current VDE library. It records the new module in the VDE database, including its facility, name, type, remark string, and other attributes. Every module belongs to some facility, either the facility you explicitly specify or the current default facility.

Modules are classified as source modules or derived modules. A *source module* is a text file (normally) which you create and modify manually with an editor. A source module can be reserved from and replaced into the VDE library with the RESERVE and REPLACE commands. The text of each source module is stored in a delta file within a CMS library in the form of a CMS ''element.''

By default, the CREATE MODULE command does not create the initial generation of a source module. Instead it records the module in the VDE database and reserves it for you in your default development stream. You must then use the REPLACE command to actually create the initial generation of the module. This should be the normal way of creating new modules once the VDE library is established. However, if you use the /INPUT or /FROM_DIRECTORY qualifier, the CREATE MODULE command creates the initial generations of the new modules; these qualifiers are useful when you initially establish your VDE library. (Note that /INPUT may not be available, depending on the setting of the queued-replacement option on the stream. If /INPUT is not accepted, you must use a subsequent REPLACE command to specify the contents of the module under creation.)

A *derived module* is a file or other entity that is produced by an automated build step, such as an object module or an executable image. A derived module cannot be reserved or replaced. In most cases, VDE automatically enters derived modules into the VDE database when the modules are produced; you seldom need to enter them with the CREATE MODULE

command. However, when a derived module needs to be entered into the database manually, you can do so with the /DERIVED qualifier to the CREATE MODULE command.

If a module is mistakenly created, one can use the DELETE MODULE command to reverse the action and delete it—the UNRESERVE command will *refuse* to delete modules that lack a first generation.

The DELETE MODULE command allows the creator of an errant module to correct simple mistakes—such as a misnamed module—without requiring unusual privileges. With the DELETE MODULE command, the creator can delete the module *prior* to the initial module replacement. DELETE MODULE will cancel the outstanding reservation on the module, will remove the module from any reservation session that it might be included in, and will prevent the module from ever appearing in the VDE library. For further information on the DELETE MODULE command, see DELETE MODULE. (The DELETE MODULE command normally requires DELMOD privilege. In this particular situation, CREMOD privilege is sufficient to permit module deletion.)

QUALIFIERS

/BINARY
/NOBINARY (default)

Specifies whether the source module is a binary file or not. The /BINARY qualifier specifies that the source module is a binary (non-text) file. Binary source files cannot be reviewed or differenced with the REVIEW REPLACEMENT command because they do not consist of ordinary ASCII text. The /NOBINARY qualifier specifies that the source module is an ordinary ASCII text file. These qualifiers are only meaningful for source modules.

Large files that undergo extensive changes at frequent intervals, or are the output of a language or application, may not be suited for storage as standard delta files as the file delta is never used, occupies a considerable amount of disk space, and grows rapidly. See the /MARKER qualifier. Examples of files well-suited for storage as marker files include most executable images, object files, Postscript files, and BACKUP savesets.

/CONCURRENT(default)
/NOCONCURRENT

Specifies whether the module can be reserved concurrently by more than one user. The /CONCURRENT qualifier allows the module to be reserved concurrently unless a user overrides this default with for a specific reservation. The /NOCONCURRENT qualifier specifies that the module cannot be reserved concurrently.

/DELETE
/NODELETE

Controls whether the **delete** attribute is set for the new facility. This attribute allows the facility to be deleted with the DELETE FACILITY command. The /DELETE qualifier sets the delete attribute so that the new facility can be deleted. The /NODELETE qualifier clears the delete attribute so that the new facility cannot be deleted unless you first reset this attribute with a MODIFY FACILITY command. Marking a facility as /NODELETE makes it harder to accidentally delete the facility.

If this qualifier is not specified, VDE uses the value of the allow-deletion library attribute. The library allow-deletion default attribute can be set or reset by using the /ALLOW_DELETE qualifier on the CREATE LIBRARY or on the MODIFY LIBRARY command.

/DERIVED
/SOURCE (default)

Specifies whether the new module is a source module or a derived module. By definition, a source module is a module that can be reserved from and replaced into the library. The generations of each such module is stored in a ''delta file'' (a CMS element). Most source modules are text files that you modify with an editor, but VDE also supports binary (non-text) source modules.

A derived module is a module that is produced from source modules or other derived modules by an automated build step. It cannot be reserved or replaced, and is often ignored during command wildcard processing. Derived modules are automatically recorded in the VDE database when they are produced; they are seldom entered into a library with the CREATE MODULE command. However, new source modules are not automatically recorded in the VDE database and must be entered into a library with the CREATE MODULE command. For this reason, the /SOURCE qualifier is the default.

/FROM_DIRECTORY=dir-spec

Specifies that modules be created for all files in a specified disk directory. The **dir-spec** parameter gives the OpenVMS directory specification of the disk directory. When this qualifier is specified, the **mod-name** parameters are omitted, and the CREATE MODULE command creates a new module in the current default facility for each file in the specified directory. Each file becomes the initial generation of the corresponding module. The **dir-spec** parameter may also consist of a wildcarded file specification, in which case VDE creates a module for each file that matches the specification. (This feature can be used to create modules for only those files in a directory that have a certain file extension, for example.) This qualifier lets you enter a large number of new modules into the VDE library with a single command, provided that the initial versions of those modules are stored in a single disk directory and that all modules belong to the same facility.

The initial generation of each module created with this qualifier is inserted into (becomes the latest generation of) the stream you specify with the /STREAM qualifier or into the default stream if you omit the /STREAM qualifier. It is also inserted into all successors of that stream.

This qualifier requires the PERFREP privilege.

/HISTORY["string"]
/NOHISTORY["string"] (default)

Specifies the CMS history string to be associated with the module and whether history information should be included in source files when modules are retrieved with the FETCH or RESERVE command. History information gives a historical record of previous changes to the module.

The /HISTORY qualifier specifies that history information should be included in source files by default when such files are retrieved with the FETCH or RESERVE command. The /NOHISTORY qualifier specifies that history information should be omitted. The **string** parameter to either qualifier specifies the default history string if the user specifies that history information should be included in a retrieved file. (If you specify the /NOHISTORY qualifier on the CREATE MODULE command, you must use the /HISTORY qualifier on the FETCH or RESERVE command to use the default history string.) If you do not specify a **string** parameter, VDE supplies a default history string that depends on the module type.

See the CMS documentation for a description of the CMS history string and its format.

/INPUT[=file-spec]

The /INPUT qualifier causes the CREATE MODULE command to create the initial generation of the new module. This qualifier is only applicable to source modules. The **file-spec** parameter gives the OpenVMS file specification of a file that contains the initial generation of the module. If the file-name part of the file specification is omitted, the file name is assumed to be the same as the module name. If the directory specification part is omitted, the file is assumed to be in your current default directory. If you omit the whole **file-spec** parameter, VDE creates the initial generation from the file in your default directory with the same name as the new module.

The initial generation of each module created with the /INPUT qualifier is inserted into (becomes the latest generation of) the stream you specify with the /STREAM qualifier or into the default stream if you omit the /STREAM qualifier. It is also inserted into all successors of that stream.

If neither /INPUT nor /FROM_DIRECTORY is specified, the CREATE MODULE command records the module in the VDE database and marks it as reserved to you. You must then use the REPLACE command to actually create the initial generation of the module. This is the preferred way of creating new modules once a VDE library has been established.

The /INPUT qualifier is rejected on any streams marked for queued replacement. To create and insert a module on a stream marked for queued replacement, omit the /INPUT qualifier, and specify the file to load into the new module via a REPLACE command.

This qualifier require the PERFREP privilege.

/LOG (default)
/NOLOG

Controls whether log messages are printed after the new modules have been created. The /LOG qualifier prints the messages and /NOLOG suppresses them. The messages indicate that the new modules have been added to the VDE library and that the database transaction has successfully committed.

/MARKER
/NOMARKER (default)

Specifies whether the source module uses a marker file or not. The /MARKER qualifier specifies that the source module uses a marker file. When a source module uses a marker file, the module is stored as an ordinary file in one of the delta-file directories. A small text file, called the

marker file, contains the full file name of that source file, and this marker file is stored in the module's delta file (CMS element). In short, the delta file contains a small text file that points to the module's actual source file. The marker-file mechanism is normally only used for large or frequently changed binary source modules, for which the delta-file mechanism may give poor performance or use excessive disk space. Executable images, object modules, Postscript files, and BACKUP savesets are all example of ''source'' modules for which the marker-file mechanism may be appropriate.

The actual source files for modules that use marker files are stored in a subdirectory for the module's facility within the VDE library's delta-file directory tree. This subdirectory must have the special directory symbol VDE$MARKER. See the description of the SET DIRECTORY command for an explanation of how to establish subdirectories and assign them directory symbols. See the description of the command SHOW DIRECTORY/DELTA for information on displaying the current marker file directory (or directories) setting.

The /NOMARKER qualifier specifies that the source module does not use a marker file and that its text is stored directly in the delta file (CMS element) for the module. Ordinary ASCII text modules should not use marker files.

Also see the /[NO]BINARY qualifier.

/NOTES[="string"]
/NONOTES[="string"] (default)

Specifies the CMS notes string to be associated with the module and whether CMS notes should be included in source files when modules are retrieved with the FETCH or RESERVE command. CMS notes indicate in what module generation each source line was created.

The /NOTES qualifier specifies that notes information should be included in source files by default when such files are retrieved with the FETCH or RESERVE command. The /NONOTES qualifier specifies that notes information should be omitted. The **string** parameter to either qualifier specifies the default notes string if the user specifies that notes information should be included in a retrieved file. (If you specify the /NONOTES qualifier on the CREATE MODULE command, you must use the /NOTES qualifier on the FETCH or RESERVE command to use the default notes string.) If you do not specify a **string** parameter, VDE supplies a default notes string that depends on the module type.

See the CMS documentation for a description of the CMS notes string and its format.

/NOTIFY=(username [,username...])

Specifies users to notify of all changes to the new module. Each **username** parameter specifies the OpenVMS username or mail address of a user who should be notified by mail message each time the specified module is replaced into the VDE library.

Each **username** parameter can specify the username of a user already recorded in the VDE database, a OpenVMS mail address with one or more node names (such as NODE::SMITH), or a mail forwarding alias, or a defined logical name bound to a list of users to notify. If you specify a logical name, keep in mind that the logical name must be defined when it is later used to send notification messages. It should thus be a group or

system logical name or a logical name that is in some other way defined for all users of the VDE library.

If you specify only one **username** parameter, you may omit the surrounding parentheses.

/POSITION=column-number

Specifies the default column number in which CMS notes are to be placed when this source module is retrieved with a FETCH or RESERVE command. The column number must be an integer in the range 1 to 511. If this qualifier is omitted, the default column number is 81. This qualifier has no effect unless notes are enabled with the /NOTES qualifier on the CREATE MODULE command or on the FETCH or RESERVE command.

See the CMS documentation for a description of CMS notes and how notes are formatted at the ends of source lines.

/PROPAGATE
/NOPROPAGATE

Specifies whether automatic change propagation is allowed for this module when it is reserved and replaced into the library. /PROPAGATE allows automatic change propagation and /NOPROPAGATE disallows change propagation. If change propagation is disallowed, VDE automatically inhibits change propagation when the module is reserved or replaced as if the /NOPROPAGATE qualifier had been specified for the RESERVE or REPLACE command. If you omit both qualifiers, the value of the facility's **nopopulate** attribute is used to set the new module's **propogate** attribute.

/REMARK="string"

Specifies a remark string to be stored for the new module. The quoted remark string, which can be up to 132 characters long, may contain any explanatory remark about the new module that you want stored in the VDE database. Typically the remark describes the purpose or content of the module.

Although this qualifier is required, you can specify an empty remark string. If you do not specify the /REMARK qualifier, VDE prompts you for a remark string.

/REVIEWER=(username [,username...])

Specifies reviewers for all changes to the new module. Each **username** parameter specifies the OpenVMS username of a user who should review all changes to the specified module. The user must already be recorded in the VDE database. When a change to the module is replaced into the VDE library, VDE notifies the user of the change by mail message. If the replacement is queued, the user should then review the queued replacement before it is performed. If the replacement is immediate, VDE notifies the user of the new module generation that was created, and the user may then inspect that generation.

If you specify only one **username** parameter, you may omit the surrounding parentheses.

/SESSION[=session-name]

Sessions are used to logically group a set of module reservations together, typically to group all modules related to a particular source code alteration or enhancement together. It allows all component modules reserved to

be treated as a single entity for subsequent replacement operations. A session also allows additional modules to be reserved and incorporated into an existing session at a later time.

If the new module is being created such that the initial generation will be created by a subsequent replacement, /SESSION specifies that VDE should add the new module reservation to the reservation session given by the **session-name** parameter.

If no session with the specified **session-name** session name currently exists, VDE will create a new reservation session under the specified session name and will incorporate the module reservation(s) into it.

If the **session-name** parameter is omitted, VDE generates a unique session name, creates a new session by that name, and adds the module reservation to it.

If the /SESSION qualifier is omitted on the CREATE MODULE command, VDE will reserve the module (as expected), and will not add the new module reservation to any session.

Sessions can be manipulated via the REPLACE, RESERVE, UNRESERVE, MODIFY SESSION, MODIFY RESERVATION, CREATE MODULE, and CANCEL SESSION commands. And modules created by CREATE MODULE (on a queued-replacement stream) and reserved via RESERVE can be combined into the same session.

/STREAM=stream-name

Specifies that the module be reserved for the stream given by the **stream-name** parameter or that the initial generation of the module be inserted into that stream. If neither /INPUT nor /FROM_DIRECTORY is specified, the /STREAM qualifier causes VDE to mark the module as reserved for the specified stream. If the /INPUT or /FROM_DIRECTORY qualifier is specified, /STREAM causes the the initial generation of the new module to become the latest generation of the module for the specified stream and all its successor streams.

If the /STREAM qualifier is omitted, the module is reserved for or inserted into your current default stream.

EXAMPLES

1

```
VDE> CREATE MODULE STGCVT.FOR /STREAM=FOO /REMARK="String conversion routines"
%VDE-I-RESERVED, generation [CODE]STGCVT.FOR reserved from stream FOO
%VDE-I-MUSTREPL, you must replace the new module to create its first generation
%VDE-I-MODADDED, module [CODE]STGCVT.FOR added to the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example creates a new source module called STGCVT.FOR. The module is created in the current default facility, facility CODE, and is reserved from the specified stream FOO. A subsequent REPLACE command (not shown) is needed to create the first generation of the module.

2

```
VDE> CREATE MODULE/NOLOG STGCVT.FOR /STREAM=FOO /REMARK="String conversion routines"
VDE>
```

Here the same module is created as in the previous example, but the /NOLOG qualifier suppresses the log messages.

3

```
VDE> CREATE MODULE STGCVT.FOR /NOLOG /INPUT=[MYDIR]FOO.FOR -
VDE>                          /REMARK="String conversion routines"
VDE>
```

In this example, the /INPUT qualifier causes the command to create the initial generation of the module from file [MYDIR]FOO.FOR. In this case, the module is not marked as reserved.

4

```
VDE> CREATE MODULE [CODE]STGCVT.FOR /REMARK="String routines", -
_VDE>              [ACCTNG]SALES.COB /REMARK="Sales accounting"
%VDE-I-RESERVED, generation [CODE]STGCVT.FOR reserved from stream MAIN
%VDE-I-MUSTREPL, you must replace the new module to create its first generation
%VDE-I-MODADDED, module [CODE]STGCVT.FOR added to the database
%VDE-I-RESERVED, generation [ACCTNG]SALES.COB reserved from stream MAIN
%VDE-I-MUSTREPL, you must replace the new module to create its first generation
%VDE-I-MODADDED, module [ACCTNG]SALES.COB added to the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example creates two new source modules, one in facility CODE and one in facility ACCTNG. Each module has its own remark string. The modules are reserved in the user's default stream, stream MAIN.

5

```
VDE> SET STREAM MAIN
VDE> SET FACILITY CODE
VDE> CREATE MODULE/FROM_DIRECTORY=[MYDIR.SOURCES]/REMARK=""
%VDE-I-GENINSERT, generation [CODE]A.REQ;1(1) inserted into stream MAIN
%VDE-I-MODADDED, module [CODE]A.REQ added to the database
%VDE-I-GENINSERT, generation [CODE]B.REQ;1(1) inserted into stream MAIN
%VDE-I-MODADDED, module [CODE]B.REQ added to the database
%VDE-I-GENINSERT, generation [CODE]C.B32;1(1) inserted into stream MAIN
%VDE-I-MODADDED, module [CODE]C.B32 added to the database
%VDE-I-GENINSERT, generation [CODE]D.B32;1(1) inserted into stream MAIN
%VDE-I-MODADDED, module [CODE]D.B32 added to the database
%VDE-I-GENINSERT, generation [CODE]E.B32;1(1) inserted into stream MAIN
%VDE-I-MODADDED, module [CODE]E.B32 added to the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example uses the /FROM_DIRECTORY qualifier to create modules for all files in a given directory. The SET STREAM command makes stream MAIN the default stream and the SET FACILITY command makes facility CODE the default facility. The CREATE MODULE command reads all files in directory [MYDIR.SOURCES] and creates a module in facility CODE for each such file. VDE creates the initial generation of each module and inserts that generation into the default stream (stream MAIN) and all successor streams. (No successor streams are defined in this example.) The log messages confirm that the initial generations have been inserted into the appropriate streams and that the modules have been added to the database.

CREATE REPLACEMENT

Creates a new queued replacement in the VDE database.

Requires RESREP privilege.

FORMAT

CREATE REPLACEMENT *rep-name [, rep-name...]*

PARAMETER

rep-name

The name of the new replacement to be created. This name can be up to 39 characters long and must follow VDE name syntax.

DESCRIPTION

When a user has reserved and modified one or more modules and then uses the REPLACE command to replace the modules into the VDE libary, VDE either replaces the modules immediately or creates a queued replacement for them. A queued replacement is a record in the VDE database that records that the modules should be replaced into the library at a future time, after the modules have been reviewed and when an authorized user enters a PERFORM REPLACEMENT command for the replacement. Whether the REPLACE command replaces the modules immediately or creates a queued replacement is determined by a qualifier to that command or by an attribute on the streams affected by the replacement.

Although queued replacements are normally created by the REPLACE command when you replace modules, you can create a queued replacement directly with the CREATE REPLACEMENT command. After you have created a queued replacement with this command, you can add module reservations to the replacement with the REPLACE command by using the /REPLACEMENT qualifier to specify the replacement name.

QUALIFIERS

/LOG (default)
/NOLOG

Controls whether log messages are printed after the new replacements are created. The /LOG qualifier causes the messages to be printed and the /NOLOG qualifier suppresses them. The messages indicate that each new replacement has been created and that the database transaction has successfully committed.

/REMARK="string"

Specifies a remark string to be stored for the new replacement. The quoted remark string, which can be up to 132 characters long, may contain any explanatory remark about the new replacement that you want stored in the VDE database.

Although this qualifier is required, you can specify an empty remark string. If you do not specify the /REMARK qualifier, VDE prompts you for a remark string.

EXAMPLES

1

```
VDE> CREATE REPLACEMENT NEWREP/REMARK="Replacement remark"
%VDE-I-REPADDED, replacement NEWREP added to the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
VDE> REPLACE/QUEUE/REPLACEMENT=NEWREP A.REQ "Fix access violation"
%VDE-I-FILCOPSTAG, file DEV$:[MYDIR]A.REQ;13 copied to staging area
%VDE-I-REPLQUE, module [CODE]A.REQ queued for replacement NEWREP
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
VDE> PERFORM REPLACEMENT NEWREP
%VDE-I-REPLACED, generation [CODE]A.REQ;8(6A2) replaced into stream V2.0
%VDE-I-REPPERF, replacement NEWREP performed
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example creates a new replacement called NEWREP and specifies a remark string for it. The REPLACE command then adds a reservation for module [CODE]A.REQ to the replacement. The subsequent PERFORM REPLACEMENT command replaces that module into the VDE library.

CREATE SCRIPT

Creates a VDE script in the VDE database.

Requires CRESCRP privilege.

FORMAT

CREATE SCRIPT *file-spec*

PARAMETER

file-spec

Specifies the OpenVMS file specification for a file that contains the text of the new script. If you omit the directory specification from the **file-spec** parameter, your current default directory is assumed. If you omit the file extension from the **file-spec** parameter, file extension VDESCRIPT is assumed.

DESCRIPTION

The CREATE SCRIPT command reads a VDE script from a file and records the script and its type in the VDE database. A **script** is a sequence of commands in the VDE script language that VDE can execute to create a DCL command file. VDE can then invoke that DCL command file in a subprocess to perform the action of the script. VDE uses scripts to perform many different kinds of actions. For example, VDE uses ''compile scripts'' to compile modules in a build job. Similarly, it uses ''new-stream scripts'' to populate the directories of a new stream with files when it performs the CREATE STREAM command. There are many other kinds of scripts, all distinguished by the commands or circumstances that cause them to be invoked.

The primary purpose of VDE scripts is to insert names and directory specifications from the VDE database into text you specify to generate DCL command files that perform various actions. A VDE script is thus a mixture of plain text and invocations of functions that retrieve names (such as module names, type names, and facility names) and directory strings (such as the names of facility subdirectories) from the VDE database.

Since you define scripts, you define how VDE should perform the corresponding actions. For example, you decide how a Pascal module is compiled because you define the Pascal compile script. Similarly, you can define a script that defines auxiliary actions of the REPLACE command. Scripts thus let you make policy decisions for your own project, while VDE provides the mechanisms that implement your policies.

The CREATE SCRIPT command only defines the new script for the current default development stream. You can reference the new script only from within the stream to which it belongs. When you want to create the same script for more than one stream, you must use the CREATE SCRIPT command for each stream. For each stream, first enter the SET STREAM command to set the development stream, then enter the CREATE SCRIPT command to create the desired script within the stream.

CREATE SCRIPT

When a new development stream is created, all script definitions from the parent stream are copied to the child stream. If modified, the script definitions then evolve independently in the two development streams.

Keywords used within scripts are documented in Script Keywords, while script functions are in Script Functions, and script types are in Script Types. For other script-related commands, also see DELETE SCRIPT, EXTRACT SCRIPT, INVOKE SCRIPT, MODIFY SCRIPT, and SHOW SCRIPT.

QUALIFIERS

/BUILD

Specifies that the script be invoked when VDE starts each process that executes a build job. The VDE$BUILD utility, which executes build jobs, invokes the script once for each process that executes a given build job. This script runs before the process executes any build steps. The script can be used to generate DCL that defines commands or logical names that are needed by subsequent build steps. BUILD scripts are optional and are not needed for build jobs to work.

/COMPILE=mod-name

Specifies that the script be invoked when VDE compiles a module that matches the **mod-name** parameter. VDE invokes the script when you enter a COMPILE command for a matching module or run a build job that contains compile steps for matching modules. The script must generate the DCL commands that actually perform the compile step.

/COPY=mod-name

Specifies that the script be invoked when VDE copies a module that matches the **mod-name** parameter. VDE invokes the script when you enter a COPY command for a matching module or run a build job that contains copy steps for matching modules. The script must generate the DCL commands that actually perform the copy step.

/CREATE_FACILITY

Specifies the script that VDE should invoke when a new facility is created. This script is optional.

/DELETE_GENERATION=mod-name

Specifies that the script be invoked when VDE deletes a stream's latest generation of a module that matches the **mod-name** parameter. VDE invokes the script when you enter a DELETE GENERATION command for a matching module. The script should perform auxiliary processing that you want for the DELETE GENERATION command, such as updating source files in a clear-copy directory. This type of script is optional and is not needed for the DELETE GENERATION command to work.

/FETCH=mod-name

Specifies that the script be invoked when VDE fetches a module that matches the **mod-name** parameter. VDE invokes the script when you enter a FETCH command for a matching module. The script should generate DCL commands that perform auxiliary processing that you want for the FETCH command, such as sending notification to interested individuals. FETCH scripts are optional and are not needed for the FETCH command to work.

/INSERT_GENERATION=mod-name

Specifies that the script be invoked when VDE inserts a generation of a module that matches the **mod-name** parameter into a development stream. VDE invokes the script when you enter an INSERT GENERATION command for a matching module. The script should perform auxiliary processing that you want for the INSERT GENERATION command, such as updating source files in a clear-copy directory. This type of script is optional and is not needed for the INSERT GENERATION command to work.

/LINK=mod-name

Specifies that the script be invoked when VDE links a module that matches the **mod-name** parameter. VDE invokes the script when you enter a LINK command for a matching module or run a build job that contains link steps for matching modules. The script must generate the DCL commands that actually perform the link step.

/LOG (default)
/NOLOG

Controls whether log messages are printed after the script is created. The /LOG qualifier causes such messages to be printed and the /NOLOG qualifier suppresses them. These messages indicate that the script has been created and that the database transaction has committed successfully.

/NEW_STREAM=fac-name

Specifies that the script be invoked for each facility that matches the **fac-name** parameter when VDE creates a new development stream. VDE invokes the script for each matching facility when you enter a CREATE STREAM command. The script must generate DCL commands that copy all stream-specific files for a given facility from the parent stream's directories to the new stream's directories. The role of NEW_STREAM scripts is to populate the new stream's directory tree with all relevant files from the parent stream.

NEW_STREAM scripts are optional and are not needed for the CREATE STREAM command to work, but they are strongly recommended because minimal system builds for the new stream cannot work unless all result files (such as object libraries and executable images) are copied from the parent stream to the new child stream.

/NOKEEP=mod-name

Specifies that the script be invoked when VDE replaces a module that matches the **mod-name** parameter. VDE invokes the script when you enter a REPLACE command with the /NOKEEP qualifier for a matching module. The script should generate DCL commands that perform the action of the /NOKEEP qualifier, such as deleting the replaced source module and perhaps also the associated object and listing files from the user's default directory. The script is run after the REPLACE command's database transaction has committed; it thus performs ''clean-up'' operations for the command. NOKEEP scripts are optional and are not needed for the REPLACE command to work. (If a NOKEEP script is not defined when a module is replaced with the /NOKEEP qualifier, VDE does not delete any files.)

/REMARK="string"

Specifies a remark string to be stored in the database for the new script. The quoted remark string, which can be up to 132 characters long, may contain any explanatory remark about the script that you want to store in the database.

/REPLACE=mod-name

Specifies that the script be invoked when VDE replaces a module that matches the **mod-name** parameter. VDE invokes the script when you enter a REPLACE command for a matching module. The script should generate DCL commands that perform auxiliary processing that you want for the REPLACE or (on a queued-replacement stream) PERFORM REPLACEMENT command, such as sending notification to interested individuals or copying object modules to object libraries. REPLACE scripts are optional and are not needed for the REPLACE command to work.

/RESERVE=mod-name

Specifies that the script be invoked when VDE reserves a module that matches the **mod-name** parameter. VDE invokes the script when you enter a RESERVE command for a matching module. The script should generate DCL commands that perform auxiliary processing that you want for the RESERVE command, such as sending notification to interested individuals. RESERVE scripts are optional and are not needed for the RESERVE command to work.

/SEGMENT=seg-num

Specifies the segment number of the new script. The **seg-num** parameter, which specifies the script's segment number, must be an integer value in the range from 1 to 20. If you omit this qualifier, VDE uses segment number 10 by default. When VDE later invokes scripts to perform some action, it generates the segments of the DCL command file from scripts ordered by their segment numbers.

/SET_FACILITY=fac-name

Specifies that the script be invoked when VDE sets a new default facility that matches the **fac-name** parameter. VDE invokes the script when you enter a SET FACILITY or SET CONTEXT command for a matching facility. The script should generate DCL commands that perform auxiliary processing that you want for the SET FACILITY or SET CONTEXT command, such as defining logical names. SET_FACILITY scripts are optional and are not needed for the SET FACILITY or SET CONTEXT command to work.

This qualifier is not yet implemented.

/SET_STREAM

Specifies that the script be invoked when VDE sets a new default stream. VDE invokes the script when you enter a SET STREAM or SET CONTEXT command. The script should generate DCL commands that perform auxiliary processing that you want for the SET STREAM or SET CONTEXT command, such as defining logical names. SET_STREAM scripts are optional and are not needed for the SET STREAM or SET CONTEXT command to work.

This qualifier is not yet implemented.

/STAGING=mod-name

Specifies that the script be invoked when VDE copies a source module that matches the **mod-name** parameter into a ''staging'' area. VDE invokes the script when you enter a REPLACE command for a matching module and the replacement is queued. For a queued replacement, VDE copies each replaced file to a staging area, from where it is actually replaced into the VDE library when another user later enters a PERFORM REPLACEMENT command. The script must generate DCL commands that copy the module into the staging area.

/UNRESERVE=mod-name

Specifies that the script be invoked when VDE unreserves a module that matches the **mod-name** parameter. VDE invokes the script when you enter an UNRESERVE command for a matching module. The script should generate DCL commands that perform auxiliary processing that you want for the UNRESERVE command, such as sending notification to interested individuals. UNRESERVE scripts are optional and are not needed for the UNRESERVE command to work.

EXAMPLES

1

```
$ TYPE [MYDIR]SCRIPT_C.VDESCRIPT
! Script to compile a C module.
!
EMIT "$ SET VERIFY"
EMIT "$ DEFINE/NOLOG SRC$ ", $DIR(SRC)
EMIT "$ DEFINE/NOLOG OBJ$ ", $DIR(OBJ)
EMIT "$ CC/DEBUG/OBJ=OBJ$:/ANA=OBJ$: SRC$:",$MODTYP
EMIT "$ VDE/DEPEND C OBJ$:",$MOD,".ANA"
EXIT
$ VDE
VDE> CREATE SCRIPT/COMPILE=[RMS]*.C [MYDIR]SCRIPT_C.VDESCRIPT
%VDE-I-SCRPADDED, script from file DEV$:[MYDIR]SCRIPT_C.VDESCRIPT;9 added to
 the database
 script to compile module [RMS]*.C (segment 10)
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

In this example, the user first types the script file [MYDIR]SCRIPT_ C.VDESCRIPT on the terminal. The script contains five EMIT commands followed by an EXIT command. The user then enters VDE and uses the CREATE SCRIPT command to enter that script file into the VDE database as the compile script for all C modules in facility RMS. The log messages confirm that the script was successfully entered into the database.

2

```
$ TYPE [MYDIR]SCRIPT_LINK_X.VDESCRIPT
! VDE script to generate the DCL commands needed to link module X.EXE.
!
EMIT "$ SET VERIFY"
EMIT "$ DEFINE OBJLIB ", $DIR(OBJ,CODE)
EMIT "$ LINK/EXE=OBJLIB:X.EXE/MAP=OBJLIB:X.MAP  -"
EMIT "        OBJLIB:OBJLIB.OLB/INCLUDE=C,OBJLIB:OBJLIB.OLB/LIBRARY"
EMIT "$ DEASSIGN OBJLIB"
$ VDE
VDE> CREATE SCRIPT/LINK=X.EXE [MYDIR]SCRIPT_C
%VDE-I-SCRPADDED, script from file DEV$:[MYDIR]SCRIPT_LINK_X.VDESCRIPT;2
 added to the database
 script to link module [CODE]X.EXE (segment 10)
%VDE-I-COMMIT, database transaction has successfully committed
VDE> SHOW SCRIPT/TEXT/LINK=X.EXE
Script to link module [CODE]X.EXE (segment 10)
   Text of script:
     1: ! script to generate the DCL commands needed to link module X.EXE.
     2: !
     3: EMIT "$ SET VERIFY"
     4: EMIT "$ DEFINE OBJLIB ", $DIR(OBJ,CODE)
     5: EMIT "$ LINK/EXE=OBJLIB:X.EXE/MAP=OBJLIB:X.MAP  -"
     6: EMIT "        OBJLIB:OBJLIB.OLB/INCLUDE=C,OBJLIB:OBJLIB.OLB/LIBRARY"
     7: EMIT "$ DEASSIGN OBJLIB"

VDE>
```

This example displays the script file SCRIPT_LINK_X.VDESCRIPT to show the text of the script. The script consists of two comment lines and five EMIT commands. The EXIT command is omitted in this case. The user then enters VDE and uses the CREATE SCRIPT command to enter the script into the VDE database. The /LINK qualifier specifies that the script links module X.EXE in the current default facility, facility CODE. By default, the script's segment number is 10. Finally, the SHOW SCRIPT command shows that the script can now be displayed directly from the database.

3

```
VDE> CREATE SCRIPT/NEW_STREAM=* SCRIPT_NEW_STRM.VDESCRIPT
%VDE-I-SCRPADDED, script from file DEV$:[MYDIR]SCRIPT_NEW_STRM.VDESCRIPT;3
 added to the database
 script to create new stream for facility * (segment 10)
%VDE-I-COMMIT, database transaction has successfully committed
VDE> SHOW SCRIPT/TEXT/NEW_STREAM=*
Script to create new stream for facility * (segment 10)
   Text of script:
     1: ! VDE script to populate the directories for a facility
     2: ! when a new stream is created.
     3: !
     4: EMIT "$ DEFINE/NOLOG OLDOBJ ", $DIR(OBJ,%CURRENT,%PARENT)
     5: EMIT "$ DEFINE/NOLOG NEWOBJ ", $DIR(OBJ,%CURRENT,%CURRENT)
     6: EMIT "$ COPY OLDOBJ:*.OLB NEWOBJ:"
     7: EMIT "$ COPY ",$DIR(SRC,%CURRENT,%PARENT),"*.H -"
     8: EMIT "       ",$DIR(SRC,%CURRENT,%CURRENT)
     9: EXIT

VDE>
```

This example creates a NEW_STREAM script that will be executed for all facilities when a new development stream is created. The CREATE SCRIPT command creates the script, and the SHOW SCRIPT command fetchs and displays the contents of the script from the database. The script generates DCL commands to copy all object libraries from the parent stream's object directory to the current stream's object directory. (The ''current'' stream is the new stream being created in this case.) The

script also copies all .H files from the parent stream's source directory for the facility to the new stream's source directory for the facility.

```
4  VDE> CREATE SCRIPT/INSERT_GENERATION=[*]*.* [MYDIR]SCRIPT_INS_GEN
   %VDE-I-SCRPADDED, script from file DEV$:[MYDIR]SCRIPT_INS_GEN.VDESCRIPT;4 added
    to the database
    script to insert generation for module [*]*.* (segment 10)
   %VDE-I-COMMIT, database transaction has successfully committed
   VDE>
   VDE> SHOW SCRIPT/TEXT/INSERT_GENERATION=[*]*.*
   Script to insert generation for module [*]*.* (segment 10)
      Text of script:
        1: ! Script for the INSERT GENERATION command.  This script fetches
        2: ! out a clear copy of the generation just inserted into a stream
        3: ! in order to keep the stream's source directory up-to-date.
        4: !
        5: FETCH $FACMODTYP, "/LOG/OUTPUT=", $DIR(SRC)
        6: EMIT "$ PURGE ", $DIR(SRC),$MODTYP

   VDE>
```

This example defines a script to be executed each time the INSERT GENERATION command is performed for any module. The SHOW SCRIPT command displays the text of the script. The script contains a FETCH command that fetches the just inserted generation into the SRC subdirectory for the affected stream and facility. This operation maintains an up-to-date copy of the source file in that directory. Because the /LOG qualifier is generated for the FETCH command, VDE will print an informational log message as the module is fetched during script execution. The script also contains an EMIT command that generates a PURGE command to be performed on the file just fetched out.

CREATE SESSION

Creates a new reservation session in the VDE database. Sessions are used to group reservations in the VDE database. Attributes of a session include the session name name and the session remark string.

Requires RESREP privilege.

FORMAT

CREATE SESSION *session-name [, session-name...]*

PARAMETER

session-name

The name of the new session to be created. This name can be up to 39 characters long and must follow VDE name syntax.

DESCRIPTION

When a user reserves one or more modules with the /SESSION qualifier to the RESERVE command, VDE creates a reservation session and makes the module reservations members of that session. A reservation session is thus a set of module reservations that belong together. Additional module reservations can be added to the session with additional RESERVE commands. When the modules are later replaced into the VDE library, the user can specify the session to be replaced (using the /SESSION qualifier to the REPLACE command) instead of listing all the modules individually. A session thus makes it easy to replace a set of related module reservations as a single unit.

Although reservation sessions are normally created by the RESERVE command, you can create a session directly with the CREATE SESSION command. This session initially contains no module reservations. You can later add module reservations to the session with the RESERVE command, using the /SESSION qualifier to specify the session name. To remove an existing non-empty session, use the UNRESERVE/SESSION command. To remove an existing empty session, use the CANCEL SESSION command.

QUALIFIERS

/LOG (default)
/NOLOG

Controls whether log messages are printed after the new sessions are created. The /LOG qualifier causes the messages to be printed and the /NOLOG qualifier suppresses them. The messages indicate that each new reservation session has been created and that the database transaction has successfully committed.

/REMARK="string"

Specifies a remark string to be stored for the new reservation session. The quoted remark string, which can be up to 132 characters long, may contain any explanatory remark about the new session that you want stored in the VDE database.

Although this qualifier is required, you can specify an empty remark string. If you do not specify the /REMARK qualifier, VDE prompts you for a remark string.

EXAMPLES

1

```
VDE> CREATE SESSION FIX-SORT-BUG /REMARK="Fix bug in sort routine"
%VDE-I-SESADDED, session FIX-SORT-BUG added to the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
VDE> RESERVE [SYS]SORT.PAS /SESSION=FIX-SORT-BUG "Fix sort bug"
%VDE-I-FETCHED, generation [SYS]SORT.PAS;4(2A2) fetched from library
%VDE-I-OUTFILE, output file is DEV$:[MYDIR]SORT.PAS;
%VDE-I-RESERVED, module generation [SYS]SORT.PAS;4(2A2) reserved from stream MAIN
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example creates a new session called FIX–SORT–BUG and specifies a remark string for it. The RESERVE command then adds a reservation for module [SYS]SORT.PAS to the session.

CREATE STEP

Defines a new build step in the VDE database. This command lets you define a copy step or specify the input and output modules of a compile or link step.

Requires CRESTEP privilege.

FORMAT

CREATE STEP *step-mod-name = mod-name [, mod-name...]*

PARAMETERS

step-mod-name

The name of the module that is compiled, copied, or linked by the build step. This name consists of a module name and a type name separated by a period (such as MOD1.PAS) and may include a facility name enclosed in square brackets (such as [FACNAM]MOD1.PAS). The module name may not contain wildcard characters. If no facility name is specified, the current default facility is assumed.

If the step is a compile step, this module is recorded as an input to the step. If the step is a link step, this module is recorded as an output of the step. If the step is a copy step, this module is recorded as the module that is copied (fetched out) by the step.

mod-name

The name of a module that you declare to be an input to or output from the build step. This name consists of a module name and a type name (for example, MOD1.OBJ) and may include a facility name enclosed in square brackets (for example, [FACNAM]MOD1.OBJ). If no facility name is specified, the current default facility is assumed.

You can declare many modules to be inputs to or outputs from the build step by specifying wildcard characters in any of the three components of the module name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Those modules whose names match the wildcard pattern are recorded as inputs to or outputs from the build step.

If an output module you specify does not already exist in the database, the CREATE STEP command adds it to the database. Input modules must already be defined in the database, however. Also, you cannot specify a source module as an output of a build step.

If you specify the /COPY qualifier to create a copy step, you must omit the **mod-name** parameters and the preceding equal sign (=). A copy step fetches a module from its CMS library, but has no inputs or outputs other than the module given by the **step-mod-name** parameter.

DESCRIPTION

To be able to perform minimal system builds, VDE records all inputs and outputs for each build step. This information is known as ‘‘build dependency information.’’ The build dependency information from all build steps is known collectively as the ‘‘build dependency graph.’’ Normally, VDE collects build dependency information automatically when you perform system builds. However, for some build steps it is not possible to collect this information automatically; some compilers and other tools do not make this information available to VDE. For such build steps, you must use the CREATE STEP command to enter the build dependency information into the VDE database.

You can define three kinds of build steps with the CREATE STEP command: compile steps, link steps, and copy steps. A *compile step* processes the **step-mod-name** module and possibly other input modules to produce one or more output modules. Typically, a compile step runs a compiler of some sort on a source module, but the exact semantics of the step are determined by the VDE script that generates the DCL commands that perform the step.

A *link step* produces the **step-mod-name** module and possibly other output modules from one or more input modules. Typically, a link step runs the linker to produce a shareable or executable image, but again the exact semantics are determined by the VDE script associated with the step. Since a script can perform any action you want, a compile step need not run a compiler and a link step need not run the linker. To VDE, the only real difference between a compile step and a link step is that a compile step is named for one of the input modules while a link step is named for one of the output modules.

A *copy step* copies the **step-mod-name** module from one place to another. Normally, a copy step is used to fetch out a source module from its delta file (CMS element) into a clear-copy file that can be used directly by subsequent build steps. For example, C programs generally include various .H files in their compile steps. A copy step is needed for each .H file to fetch it from its CMS library into a text file that subsequent compile steps can use. Likewise, linker options files (.OPT files) must be fetched out before you can use them for subsequent link steps. You must create a copy step for any module that must be fetched into the VDE library’s directory tree during builds but which does not need any separate processing to produce other output modules. For a copy step, the specified module can be seen as both the input and the output of the step.

The CREATE STEP command deletes all dependency information for the build step before recording the inputs and outputs you specify. Thus, you can use the CREATE STEP command to completely replace the dependency information for a build step. To change only some of the dependency information for a step, use the MODIFY STEP command.

Build dependency information is always specific to the default development stream. Different development streams in a software system contain different versions of the code and generally have different build dependency graphs.

To define the same build step for more than one development stream, you must use the CREATE STEP command for each stream. For each stream, first enter the SET STREAM command to set the development stream, then enter the CREATE STEP command to enter the desired build-step dependencies into the database.

When a new development stream is created, the build dependency graph from the parent stream is copied to the child stream. The build dependency graph then evolves independently in the two streams.

QUALIFIERS

/COMPILE (default)
/COPY
/LINK

Specifies whether the build step is a compile step, copy step, or link step. The /COMPILE qualifier causes VDE to mark the step as a compile step and to record the **step-mod-name** parameter as an input module to the compile step. The /COPY qualifier causes VDE to mark the step as a copy step and to record the **step-mod-name** parameter as the module that is copied by the step. The /LINK qualifier causes VDE to mark the step as a link step and to record the **step-mod-name** parameter as an output module of the link step.

Of /COMPILE, /COPY and /LINK, /COMPILE is the default.

/INPUT (default)
/OUTPUT

Specifies whether the **mod-name** parameters are recorded as inputs to or outputs from the build step. The /INPUT qualifier records modules as inputs to the build step and the /OUTPUT qualifier records them as outputs of the step. If no qualifier is specified, all modules are assumed to be inputs.

To set the default for the command as a whole, specify the qualifier after the CREATE STEP keywords. To override that default for a single module, specify the qualifier after the **mod-name** parameter you choose to mark as an input or output.

Of /INPUT and /OUTPUT, /INPUT is the default.

/LOG (default)
/NOLOG

Controls whether log messages are printed after the build step is recorded in the database. The /LOG qualifier causes the messages to be printed and the /NOLOG qualifier suppresses them. The messages indicate that the build step has been created, that the specified modules are inputs or outputs for the step, and that the database transaction has committed successfully.

EXAMPLES

1

```
VDE> CREATE STEP FOO.MAR = FOO.$OBJ/OUTPUT
%VDE-I-COMSTEPADDED, compile step for [CODE]FOO.BAR added to database
%VDE-I-MODINSTEP, module [CODE]FOO.MAR is an input to the build step
%VDE-I-MODOUTSTEP, module [CODE]FOO.$OBJ is an output from the build step
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This command creates a build step definition for the compilation of module FOO.MAR. The step is a compile step because the /COMPILE qualifier is the default. FOO.MAR is recorded as an input to the step and module FOO.$OBJ is recorded as an output from the step, as the log messages show.

2

```
VDE> CREATE STEP/LINK MOD1.EXE = MOD1.$OBJ, MOD2.$OBJ, FOO.$OBJ
%VDE-I-LNKSTEPADDED, link step for [CODE]MOD1.EXE added to database
%VDE-I-MODOUTSTEP, module [CODE]MOD1.EXE is an output from the build step
%VDE-I-MODINSTEP, module [CODE]MOD1.$OBJ is an input to the build step
%VDE-I-MODINSTEP, module [CODE]MOD2.$OBJ is an input to the build step
%VDE-I-MODINSTEP, module [CODE]FOO.$OBJ is an input to the build step
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This command creates the build dependency information for the build step that links module MOD1.EXE. Module MOD1.EXE is the output of the step and modules MOD1.$OBJ, MOD2.$OBJ, and FOO.$OBJ are the inputs.

3

```
VDE> CREATE STEP/COMPILE/OUTPUT [CODE]MOD4.FOR = MOD4.$OBJ, [CODE]MOD4.LIS
%VDE-I-COMSTEPADDED, compile step for [CODE]MOD4.FOR added to database
%VDE-I-MODINSTEP, module [CODE]MOD4.FOR is an input to the build step
%VDE-I-MODOUTSTEP, module [CODE]MOD4.$OBJ is an output from the build step
%VDE-I-MODOUTSTEP, module [CODE]MOD4.LIS is an output from the build step
%VDE-I-COMMIT, database transaction has successfully committed
VDE> SHOW STEP/FULL MOD4.FOR
Compile Step for module [CODE]MOD4.FOR
   Dependency links for stream V2.1 follow
   Inputs to the build step:
      Module [CODE]MOD4.FOR
   Outputs from the build step:
      Module [CODE]MOD4.$OBJ
      Module [CODE]MOD4.LIS
   Build step has 1 inputs and 2 outputs

VDE>
```

The CREATE STEP command defines the compile step for module MOD4.FOR in facility CODE as having one input module (MOD4.FOR) and two output modules (MOD4.$OBJ and MOD4.LIS). The SHOW STEP command confirms that these build dependencies have been entered into the VDE database.

4

```
VDE> CREATE STEP/COPY [CODE]MOD5.H
%VDE-I-COPSTEPADDED, copy step for [CODE]MOD5.H added to database
%VDE-I-MODINSTEP, module [CODE]MOD5.H is an input to the build step
%VDE-I-MODOUTSTEP, module [CODE]MOD5.H is an output from the build step
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example creates a copy step for module MOD5.H in facility CODE. VDE records the specified module as both the input and the output of the

step since a copy step simply copies the specified module from one form to another (normally from its delta-file form to text-file form).

CREATE STREAM

Creates a development stream in the current VDE library.

Requires CRESTRM privilege.

FORMAT

CREATE STREAM *stream-name [, stream-name...]*

PARAMETER

stream-name

The name of a stream to be created. This name can be up to 39 characters long. It must follow VDE name syntax. Unlike other VDE names, the stream name can contain periods (.). The use of the facility prefix *VDE$* is expressly prohibited.

DESCRIPTION

A development stream (or simply, a ''stream'') is any separately evolving version of your software system. A stream may correspond to a previous or future version of your system or to a base-level of the system. It may correspond to any instance of the system that evolves separately from the rest of the system. Each stream of your software system captures a different state of the system, and that state may evolve independently for each stream.

Each stream of your software system contains its own versions of all modules in the system, and you may change these modules independently in each stream. Each stream also contains its own versions of various database tables, such as group definitions and module dependency information. This information may also evolve independently for each stream.

When you create a stream, the CREATE STREAM command first creates an entry for the new stream in the VDE database. It then copies all stream-specific information from the parent stream (by default, the default stream) to the new stream (the child stream). This information includes directory-creation rules, group definitions, the build dependency graph, and information about what module generations belong to the parent stream. In addition, the CREATE STREAM command executes scripts which may copy files from the parent stream's disk directories to the child stream's directories.

There are several key attributes to VDE module propogation controlled by the CREATE STREAM command, including the sequence of streams on the various lines of descent, and the presence of the particular stream on the main line of development or on a variant line of descent. The former is controlled by the parentage and successor settings, and the latter is controlled by the mainline setting.

Note: The CREATE STREAM command can and does perform specific and potentially non-reversable modifications to the internal VDE data associated with *other* streams present in the VDE database, particularly around the /MAIN and /NOMAIN settings, and around

the /PARENT and /SUCCESSOR settings. The CREATE STREAM command should be used with caution.

The explicit specification of the appropriate value for the /[NO]MAIN qualifier (/MAIN or /NOMAIN) on a CREATE STREAM command is strongly recommended.

QUALIFIERS

/CMS_CLASS
/NOCMS_CLASS(default)

Specifies whether a CMS class with the same name as the stream should be maintained in each facility's CMS library. The /CMS_CLASS qualifier causes VDE to create a CMS class for the stream in each facility's CMS library and to then maintain that class when the REPLACE command and certain other commands change what module generations belong to the stream. The /NOCMS_CLASS qualifier causes VDE to not maintain a CMS class for the stream.

The /CMS_CLASS qualifier interacts with the /INIT_CMS_CLASS and /NOINIT_CMS_CLASS qualifiers; see the descriptions of these qualifiers for details.

/CONFERENCE=file-spec
/NOCONFERENCE(default)

Specifies whether a OpenVMS Notes conference for replacement information should be associated with the new stream. The /CONFERENCE qualifier specifies that the notes conference file specification given by the **file-spec** parameter should be used to record all replacements into the stream. VDE then enters a new note into the conference for each queued or immediate replacement and a reply to that note when a queued replacement is actually performed. The /NOCONFERENCE qualifier specifies that no such notes conference should be used.

Under normal circumstances, just the file name should be specified in the **file-spec** parameter. VDE applies defaults to the notes conference file specification based on the translation of the logical name VDE$NOTES_ LIBRARY. If this logical name is not defined, the default file specification ''NOTES$LIBRARY:.NOTE'' is applied to the **file-spec** parameter.

/DEFER
/NODEFER (default)

Specifies whether the creation of the disk directory structure for the stream should be postponed. The /DEFER qualifier postpones creation of the disk directory structure and the /NODEFER qualifier causes the disk directory structure to be created immediately. Use the /DEFER qualifier when the default directory structure is inappropriate for the new stream.

If the /NODEFER qualifier is specified (or /DEFER is omitted), the CREATE STREAM command creates the disk directory structure for the stream. This structure includes a stream root directory, a root directory for each facility, and facility subdirectories that hold the files that are the contents of the facility in the new stream. The directory structure is determined by the SET DIRECTORY commands entered for the new stream (or inherited from the parent stream) or by VDE default rules. Once VDE has created the new stream's directory structure for a given

facility, VDE runs the stream's new-stream script for that facility, provided such a script is defined. Depending on how it is defined, this script may populate the directories for that facility with source modules and other files.

If the /DEFER qualifier is specified, the CREATE STREAM command does not create a directory structure for the new stream. If you use the /DEFER qualifier, you can enter SET DIRECTORY commands to specify the structure of the disk directories. Then use the CREATE DIRECTORY_ TREE command to create the directory structure on disk. The CREATE DIRECTORY_TREE command uses VDE defaults and the rules you specified with the SET DIRECTORY commands to create the new directory structure.

/DELETE
/NODELETE

Controls whether the **delete** attribute is set for the new stream. This attribute allows the stream to be deleted with the DELETE STREAM command. The /DELETE qualifier sets the delete attribute so that the new stream can be deleted. The /NODELETE qualifier clears the delete attribute so that the new stream cannot be deleted unless you first reset this attribute with a MODIFY STREAM command. This makes it harder to accidentally delete a stream. If you specify neither qualifier, VDE uses the value of the allow-deletion library attribute as the initial value of the stream's delete attribute. This library attribute is set with the /ALLOW_ DELETE qualifier to the CREATE and MODIFY LIBRARY commands.

/DIRECTORY=dir-spec

Specifies the name of the root directory of the stream. The root directory normally contains a subdirectory for each facility. Each subdirectory has additional subdirectories for the sources and derived files of the facility and stream. (To modify the default directory structure, first use the CREATE STREAM command with the /DEFER qualifier then use the SET DIRECTORY and CREATE DIRECTORY_TREE commands to create the directory structure you want.) Use the OpenVMS directory specification format to specify the **dir-spec** parameter.

/INIT_CMS_CLASS
/NOINIT_CMS_CLASS

Specifies whether VDE initializes the CMS class for the stream in each facility's CMS library when the /CMS_CLASS qualifier specifies that such a class should be maintained. /INIT_CMS_CLASS causes the CREATE STREAM or MODIFY STREAM command with the /CMS_CLASS qualifier to create and populate a CMS class with the same name as the stream in each facility. /NOINIT_CMS_CLASS causes these commands to not create or populate such CMS classes. /NOINIT_CMS_CLASS is useful for very large VDE libraries where the initialization of CMS classes take an excessively long time. For such libraries, it is better to use a separate command procedure to initialize the desired CMS classes; such a procedure can process the facilities of your library in parallel for faster completion.

If you specify neither of these qualifiers, VDE uses the initialization attribute of the new stream's parent stream. By default, the initial stream for the VDE library (stream MAIN) is created with initialization enabled (in effect, /INIT_CMS_CLASS is the default). All other streams will inherit this attribute unless you change it with an explicit /NOINIT_CMS_CLASS qualifier.

Unless you have existing command procedures or programs that depend on the presence of CMS classes, Digital recommends that you not maintain CMS classes for the streams in your VDE library. Maintaining CMS classes is quite slow, especially when creating new streams, and VDE does not need CMS classes for any of its functions.

/LOG (default)
/NOLOG

Controls whether log messages are printed after new streams are created. The /LOG qualifier causes the messages to be printed and the /NOLOG qualifier suppresses them. The messages indicate that each new stream has been created, that its disk directory structure has been created, and that the database transaction has successfully committed.

/MAIN
/NOMAIN (default)

Specifies whether the new stream is part of the main development thread for your project. The /MAIN qualifier says that the new stream is part of the main development thread for the project and is expected to be in use for a relatively long time, and is expected to be in use after the parent stream has been closed. VDE assigns the shortest possible CMS ''generation expressions'' to new generations in a main stream. The /NOMAIN qualifier says that the new stream is not part of the project's main development thread and is expected to have a shorter future lifetime than the parent stream. In this case, VDE may assign longer CMS generation expressions (using CMS ''variant letters'') to future module generations in the stream.

You normally use the /MAIN qualifier when you create a stream for your project that will be the future continuation of the parent stream. Creating stream V2.0 from stream V1.0 is an example since development of Version 1.0 of your project will cease as Version 2.0 carries the project into the future. You use the /NOMAIN qualifier when you create base levels or other streams that are expected to have short lifetimes compared to the parent stream.

If you specify the /MAIN qualifier, the new stream automatically becomes a successor of the parent stream so that changes are propagated from the parent to the new stream. If you specify the /MAIN qualifier but you do not specify the /SUCCESSOR or /NOSUCCESSOR qualifier, the main streams that are successors of the parent stream are removed as successors of the parent stream and instead become successors of the new stream. The effect is that the new stream is inserted into the successor chain after the parent stream. If you specify the /NOMAIN qualifier, no such insertion takes place.

For information on changing the main line of descent attribute specific to a particular module generation, see the documentation on the MODIFY GENERATION/MAIN_STREAM command in MODIFY GENERATION.

Specification: Explicit specification of /[NO]MAIN is not currently required, but explicit specification of this qualifier may be required in a future release of VDE. Explicit specification of this qualifier—either as /MAIN or as /NOMAIN—is highly recommended.

/NOGENERATIONS

Specifies that the new stream should not be populated with the latest generations of the parent stream. If you use this qualifier, you get a new stream containing no generations. If you omit this qualifier, VDE populates the new stream with the latest generations of the parent stream.

/NOTIFY=(username [,username...])

Specifies users to notify of all changes replaced into the new stream. Each **username** parameter specifies the OpenVMS username or mail address of a user who should be notified by mail message each time a module is replaced into the stream, and each time a new facility is created in the stream.

Each **username** parameter can specify the username of a user already recorded in the VDE database, a OpenVMS mail address with one or more node names (such as NODE::SMITH), a mail forwarding alias, or a defined logical name bound to a list of users to notify. If you specify a logical name, keep in mind that the logical name must be defined when it is later used to send notification messages. It should thus be a group or system logical name or a logical name that is in some other way defined for all users of the VDE library.

If you specify only one **username** parameter, you may omit the surrounding parentheses.

/OPEN (default)
/FROZEN
/CLOSED

Determines the status of the new stream: open, frozen, or permanently closed. If you specify the /OPEN qualifier, all users can reserve and replace source modules from the stream and thereby modify the contents of the stream.

If you specify the /FROZEN qualifier, all users may reserve modules from the stream. However, a user must have the FROZEN privilege to replace modules into the stream; non-privileged users cannot change the contents of the stream. You can later change the status to open with the MODIFY STREAM command, so the stream is not permanently frozen.

If you specify the /CLOSED qualifier, modules cannot be reserved from or replaced into the stream. This status cannot be changed with a subsequent MODIFY STREAM command unless you have the UNCLOSE privilege. Use the /CLOSED qualifier when you never want the contents of the stream to change again, for example, when capturing a base-level of your software system or when capturing the final state of a released version of your software.

Of /OPEN, /FROZEN and /CLOSED, /OPEN is the default.

/OWNER=username

Specifies the owner of the new stream. The **username** parameter specifies the OpenVMS username of a user who should be considered to be the ‘‘owner’’ of the stream. VDE assigns no special meaning to the owner of a stream, but the SHOW STREAM command displays the owner. The owner could thus be the person to contact with questions about the stream.

If you omit this qualifier, VDE records no owner for the stream.

/PARENT=stream-name

Specifies the parent of the new stream. If this qualifier is specified, the **stream-name** parameter gives the name of the development stream that should be the parent of the new stream being created. If this qualifier is omitted, the default stream becomes the parent of the new stream.

Correct specification of this qualifier is of central importance when uploading facilities and streams from an existing set of CMS libraries. See the CREATE FACILITY/FROM_CMS_LIBRARY command.

Specification of /PARENT is required.

/POPULATE (default)
/NOPOPULATE

Specifies whether the stream is allowed to be populated with generations from another stream using the POPULATE STREAM command. /POPULATE allows subsequent use of the POPULATE STREAM command to populate the stream and /NOPOPULATE disallows such use.

/PROCESS_COUNT=proc-count

Specifies the default number of processes to execute build jobs for the new stream. This parameter determines the degree of parallelism for the stream's build jobs. See the description of the BUILD command for a detailed description of this qualifier.

/QUEUE=(queue-name [,queue-name...])

Specifies the default batch queues to which build jobs for the new stream are submitted. See the description of the BUILD command for a detailed description of this qualifier and for a description of its interaction with the /PROCESS_COUNT qualifier.

/REMARK="string"

Specifies a remark string to be stored for the new stream. The quoted remark string, which can be up to 132 characters long, may contain any explanatory remark about the new stream that you want stored in the VDE database. Typically the remark indicates the version or base-level of your system.

Although this qualifier is required, you can specify an empty remark string. If you do not specify the /REMARK qualifier, VDE prompts you for a remark string.

/REPLACEMENT=keyword

Controls whether the REPLACE command permits immediate replacements into the new stream or requires queued replacements. If the **keyword** parameter is IMMEDIATE, the REPLACE command will immediately replace modules into the stream by default. If the parameter is QUEUE, the REPLACE command will always queue replacements for the stream so that modules are not replaced into the stream until a privileged user enters a PERFORM REPLACEMENT command. If you omit this qualifier, VDE permits immediate replacements into the new stream. (See the descriptions of the /IMMEDIATE and /QUEUE qualifiers to the REPLACE command for more information.)

/REVIEWER=(username [,username...])

Specifies reviewers for all changes replaced into the new stream. Each **username** parameter specifies the OpenVMS username of a user who should review all changes to the specified stream. The user must already be recorded in the VDE database. When a module is replaced into the stream, VDE notifies the user of the replacement by mail message. If the replacement is queued, the user should then review the queued replacement before it is performed. If the replacement is immediate, VDE notifies the user of the new module generation that was created, and the user may then inspect that generation.

If you specify only one **username** parameter, you may omit the surrounding parentheses.

/STEP_DELAY=time-interval

Specifies the default step-delay time interval used by build jobs for the new stream. See the description of the BUILD command for a detailed description of this qualifier.

/SUCCESSOR=(succ-name [, succ-name...])
/NOSUCCESSOR (default)

The /SUCCESSOR qualifier specifies successors to the stream being created. A successor to a stream is another stream to which source changes are propagated by the REPLACE command. Each **succ-name** parameter specifies the name of a stream that should be a successor to the new stream given by the **stream-name** parameter.

The /NOSUCCESSOR qualifier specifies that the new stream does not have any successors.

If both the /SUCCESSOR and /NOSUCCESSOR qualifiers are omitted, the new stream has no successors unless you specify the /MAIN qualifier. See the description of the /MAIN qualifier for the special rules that apply in this case.

/TIMEOUT=time-interval

Specifies the default timeout interval used by build jobs for the new stream. See the description of the BUILD command for a detailed description of this qualifier.

EXAMPLES

1
```
VDE> CREATE STREAM V5.2-BL3/REMARK="Base Level 3 of V5.2"/NOLOG
VDE>
```

This example creates a new stream called V5.2-BL3. The remark string indicates that this is base level 3 of version 5.2 of the user's software system. The /NOLOG qualifier suppresses all log messages.

2

```
VDE> CREATE STREAM V5.3 /REMARK="Version 5.3"
Creating directory tree for stream V5.3:
   Root directory DEV$:[PROJDIR.V5_3] has been created
   Facility ACCTNG root directory DEV$:[PROJDIR.V5_3.ACCTNG] has been created
      Subdirectory DEV$:[PROJDIR.V5_3.ACCTNG.SRC] has been created
      Subdirectory DEV$:[PROJDIR.V5_3.ACCTNG.OBJ] has been created
      Subdirectory DEV$:[PROJDIR.V5_3.ACCTNG.VDE$COM] has been created
      Subdirectory DEV$:[PROJDIR.V5_3.ACCTNG.VDE$LOG] has been created
   Facility CODE root directory DEV$:[PROJDIR.V5_3.CODE] has been created
      Subdirectory DEV$:[PROJDIR.V5_3.CODE.SRC] has been created
      Subdirectory DEV$:[PROJDIR.V5_3.CODE.OBJ] has been created
      Subdirectory DEV$:[PROJDIR.V5_3.CODE.VDE$COM] has been created
      Subdirectory DEV$:[PROJDIR.V5_3.CODE.VDE$LOG] has been created
   Facility RMS root directory DEV$:[PROJDIR.V5_3.RMS] has been created
      Subdirectory DEV$:[PROJDIR.V5_3.RMS.SRC] has been created
      Subdirectory DEV$:[PROJDIR.V5_3.RMS.OBJ] has been created
      Subdirectory DEV$:[PROJDIR.V5_3.RMS.VDE$COM] has been created
      Subdirectory DEV$:[PROJDIR.V5_3.RMS.VDE$LOG] has been created
   Facility SMG root directory DEV$:[PROJDIR.V5_3.SMG] has been created
      Subdirectory DEV$:[PROJDIR.V5_3.SMG.SRC] has been created
      Subdirectory DEV$:[PROJDIR.V5_3.SMG.OBJ] has been created
      Subdirectory DEV$:[PROJDIR.V5_3.SMG.VDE$COM] has been created
      Subdirectory DEV$:[PROJDIR.V5_3.SMG.VDE$LOG] has been created
   All directories for stream V5.3now exist

%VDE-I-STRMADDED, stream V5.3 added to the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example creates a new stream called V5.3, representing version 5.3 of the user software system. The log messages indicate that the directories for the new stream have been created, that the stream has been added to the database, and that the database transaction has successfully completed.

CREATE USER

Creates a database record for a new user of the current VDE library.

Requires CREUSR privilege.

FORMAT

CREATE USER *username [, username...]*

PARAMETER

username

The OpenVMS username of the user to be added to the VDE database. This name can be up to 12 characters long and must follow VDE name syntax.

DESCRIPTION

The CREATE USER command creates the database record for a new user of the current VDE library.

Each user of a VDE library must have a record in the database of that library. This record contains the user's OpenVMS username, full name string, authorized and default privileges, and remark string. After the record has been created, the user has access to the VDE library as constrained by the specified privileges. A user who does not have a record in a given VDE database cannot use the VDE Utility to gain access to the database or the corresponding library.

QUALIFIERS

/AUTH_PRIVILEGES=(priv [,priv...])

Specifies the new user's authorized privileges, which are the privileges the user is allowed to set with the VDE command SET PRIVILEGE. Each parameter to this qualifier names a single privilege. If only one privilege is specified, the parentheses may be omitted. If this qualifier is not specified, VDE assigns the standard authorized privileges of the library to the new user.

For a list of privileges, see Table 3–5 in SET PRIVILEGES.

/DEF_PRIVILEGES=(priv [,priv...])

Specifies the new user's default privileges, which are the privileges enabled each time the user starts a new VDE session. Each parameter to this qualifier names a single privilege. Each privilege specified becomes an authorized privilege as well as a default privilege, even if it has not been listed with the /AUTH_PRIVILEGES qualifier. If only one privilege is specified, the parentheses may be omitted. If the /DEF_PRIVILEGES qualifier is not specified, VDE assigns the standard default privileges of the library to the new user.

For a list of privileges, see Table 3–5 in SET PRIVILEGES.

/LOG (default)
/NOLOG

Controls whether log messages are printed after users have been added to the database. The /LOG qualifier causes the messages to be printed and /NOLOG suppresses them. The messages indicate that users have been added and that the database transaction has successfully committed.

/NAME="string"

Specifies the full name of the new user. The quoted name string, which can be up to 40 characters long, should contain the full name (first name, middle initials, and last name) of the new user.

This qualifier is required. You can specify an empty name string but this is not recommended. If you do not specify the /NAME qualifier, VDE prompts you for a name string.

/NOTIFY=(username [,username...])

Specifies users to notify of all code changes replaced into the VDE library by the new user. Each **username** parameter specifies the OpenVMS username or mail address of a user who should be notified by mail message each time the new user created by this command replaces a module into the VDE library.

Each **username** parameter can specify the username of a user already recorded in the VDE database, a OpenVMS mail address with one or more node names (such as NODE::SMITH), a mail forwarding alias, or a defined logical name bound to a list of users to notify. If you specify a logical name, keep in mind that the logical name must be defined when it is later used to send notification messages. It should thus be a group or system logical name or a logical name that is in some other way defined for all users of the VDE library.

If you specify only one **username** parameter, you may omit the surrounding parentheses.

/REMARK="string"

Specifies a remark string to be stored for the new user. The quoted remark string, which can be up to 132 characters long, may contain any explanatory remark about the new user that you want stored in the VDE database. Typically the remark describes the user's affiliation or function within the organization.

This qualifier is required although you can specify an empty remark string. If you do not specify the /REMARK qualifier, VDE prompts you for a remark string.

/REVIEWER=(username [,username...])

Specifies reviewers for all code changes replaced into the VDE library by the new user. Each **username** parameter specifies the OpenVMS username of a user who should review all changes created by the new user. The reviewer must already be recorded in the VDE database. When the new user created by this command replaces a module into the VDE library, VDE notifies the reviewer of the change by mail message. If the replacement is queued, the reviewer should then review the queued replacement before it is performed. If the replacement is immediate, VDE notifies the reviewer of the new module generation that was created, and he or she may then inspect that generation. This qualifier can be used to set up ''buddy systems'' where two users always review each other's code.

If you specify only one **username** parameter, you may omit the surrounding parentheses.

EXAMPLES

1

```
VDE> CREATE USER SMITH NAME="Jane Smith"/REMARK="OpenVMS Developer"
%VDE-I-USERADDED, user SMITH added to the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This command creates a database record for user SMITH. The /NAME qualifier specifies Jane Smith's full name and the /REMARK qualifier identifies her job. The example shows the informational log messages that the /LOG qualifier produces by default.

2

```
VDE> CREATE USER DOE /NAME="John X. Doe"/REMARK="OpenVMS Writer" -
_VDE>       /AUTH_PRIVILEGES=(BUILD,CREGRP,CREMOD) -
_VDE>       /DEF_PRIVILEGES=(BUILD,CREFAC) /NOLOG
VDE>
```

In this example, user DOE is added to the database. In addition to specifying the full name and a remark for John Doe, this command gives his authorized and default VDE privileges. Note that the CREFAC privilege becomes an authorized privilege as well as a default privilege even though it is not listed by the /AUTH_PRIVILEGES qualifier. In this case, the /NOLOG qualifier suppresses the log messages.

3

```
VDE> CREATE USER/REMARK="OpenVMS Developer" -
_VDE>       SMITH /NAME="Jane Smith", -
_VDE>       JONES /NAME="Frank Jones"
%VDE-I-USERADDED, user SMITH added to the database
%VDE-I-USERADDED, user JONES added to the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example illustrates the creation of database records for multiple users with a single CREATE USER command. This command adds users SMITH and JONES to the database. Both get the same remark string because the remark is specified immediately after the CREATE USER command. They get separate full name strings and both get the standard authorized and default privileges of the current library.

DEFINE/COMMAND

Defines a command abbreviation by associating a symbol name with a list of one or more VDE commands.

FORMAT

DEFINE/COMMAND *symbol-name="command-string"*

PARAMETERS

symbol-name

Specifies the symbol name that you want associated with a list of VDE commands. The symbol can be up to 255 characters long and can include letters, digits, underscores, and dollar signs. It cannot start with a digit. If *symbol-name* is already defined, the new definition replaces the old definition.

command-string

Specifies the string of VDE commands to associate with the symbol name. The command string must be enclosed in quotation marks and consists of the first part of a VDE command, a whole command, or a list of commands separated by semicolons.

DESCRIPTION

The DEFINE/COMMAND command associates a symbol name with a VDE command string. The command string consists of zero or more VDE commands separated by semicolons and must end with a partial or full VDE command.

When the symbol name appears at the beginning of a VDE command, the symbol name is replaced by its associated command string. The resulting string is then parsed and executed as a normal command line by VDE. DEFINE/COMMAND thus gives you the same command abbreviation capability as the symbol assignment statement (symbol-name := command-string) in DCL. Symbol substitution only happens once at the beginning of each command. A symbol's associated command string can therefore not begin with another command symbol.

To display the current symbol definitions created by DEFINE/COMMAND, use the SHOW SYMBOL command.

QUALIFIERS

None.

EXAMPLES

1

```
VDE> DEFINE/COMMAND SM = "SHOW MODULE"
VDE> SM MOD1
Module [FACIL]MOD1.MAR
VDE>
```

This DEFINE/COMMAND command associates the symbol SM with the VDE command ‘‘SHOW MODULE’’. When the SM command is used, SM is expanded to the equivalent command string. The module name MOD1 is then concatenated to that string to form the full command SHOW MODULE MOD1, which VDE then executes.

2

```
VDE> DEFINE/COMMAND STRMFAC = "SET STREAM V5.3; SET FACILITY RMS"
VDE> STRMFAC
VDE> SHOW DEFAULTS
Library LIBNAME in root directory DEV$:[PROJDIR]
Default directory is DEV$:[SMITH]
Current session context:
   Default stream is V5.3
   Default facility is RMS
VDE>
VDE>
```

This example shows that a symbol can be associated with multiple commands. Here symbol STRMFAC is associated with both a SET STREAM command and a SET FACILITY command, where the two commands are separated by a semicolon. When the STRMFAC command is entered, the command string is substituted for the symbol name and the SET STREAM and SET FACILITY commands are executed. The output from a subsequent SHOW DEFAULTS command shows that the default stream and the default facility have been set by the STRMFAC command.

DEFINE/KEY

Defines a key to execute a VDE command. A key definition lets you to enter a command by pressing a key rather than typing the command. The /KEY qualifier is required and must precede any other qualifiers.

FORMAT

DEFINE/KEY *key-name string*

PARAMETERS

key-name

Specifies the name of the key you are defining. Table 3–1 contains a list of common keys on common keyboards.

Table 3–1 Key Names

Key-name	VT100	LK201,LK401	Default VDE Usage
PF1	PF1	PF1	Sets GOLD state
PF2	PF2	PF2	Gets Help
PF3	PF3	PF3	none
PF4	PF4	PF4	Sets BLUE state
KP0, KP1-KP9	keypad 0-9	keypad 0-9	None
PERIOD	period key	period key	None
COMMA	comma key	comma key	None
MINUS	minus key	minus key	None
ENTER	ENTER key	ENTER key	None
E1,E2	-	Find,Insert Here	None
E3,E4	-	Remove,Select	None
E5	-	Previous Screen	None
E6	-	Next Screen	None
HELP	-	Help/F15	Gets Help
DO	-	Do/F16	None
F17-F20	-	F17-F20	Sets defaults to libraries

Note: You cannot redefine the arrow keys or function keys F1 through F14.

string

Specifies the string that is entered when you press the defined key. **String** can be any VDE command, such as SHOW DEFAULTS or SET CONTEXT. To preserve spaces and lowercase characters, enclose the string in quotation marks.

DESCRIPTION

The DEFINE/KEY command, on certain terminals, enables you to assign definitions to keys other than those on the main keyboard. The command can be used on a variety of terminals including the VT340, VT510 and the DECterm terminal emulator.

The string parameter to the DEFINE/KEY command can contain different types of information and usually consists of an entire VDE command. For example, you can assign the SHOW DEFAULTS command to KP0. Then, when you press zero on the keypad, VDE displays your current default settings. Other definitions can consist of text strings to be appended to command lines. When you define a key to insert a text string, use the /NOTERMINATE qualifier so you can continue typing more data after the string has been inserted.

In most instances you will want to use the echo feature. The /ECHO qualifier displays the key definition on the screen each time you press the key. The default setting is /ECHO.

You can create as many definitions for a key as you want as long as you specify a new context for each definition. This new context is called a *state*. To create additional key definitions you must use two steps. First, using the /SET_STATE qualifier, you must define a key that establishes a state for the definition you want to create. Then you must create the new definition for your key and specify its state using the /IF_STATE qualifier.

State names can include letters, digits, dollar signs, and underscores. Create state names that are easy to remember and type and that remind you of the types of definitions you created. For example, you can create a state called SETSHOW. The key definitions for this state might refer to various VDE SET and SHOW commands.

Like various other utilities on OpenVMS, VDE supplies three built-in states called DEFAULT, GOLD, and BLUE. You can assign different definitions to keys used in combination with the GOLD and BLUE keys by using the /IF_STATE qualifier.

By convention, the GOLD state is selected by pressing the PF1 key, and the BLUE state is seected by pressing the PF4 key.

The SET KEY command changes the keypad state. Use the SHOW KEY command to display key definitions and states.

QUALIFIERS

/ECHO (default)
/NOECHO

Specifies whether the command line is displayed on your screen after you press the defined key. You cannot define a key specifying both /NOECHO and /NOTERMINATE.

/IF_STATE=(state-name,...)
/NOIF_STATE (default)

Specifies a list of states, any one of which must be set to enable the specified key definition. (By including several state names, you can define a key to have the same function in all the specified states.) If you use the /NOIF_STATE qualifier or omit the /IF_STATE qualifier, the current

state at the time you define the key must be set when you use the key definition. The state name can include letters, digits, dollar signs, and underscores. States are established with the /SET_STATE qualifier on another key definition or with the SET KEY command. If you specify only one state name, you can omit the parentheses.

/LOCK_STATE
/NOLOCK_STATE (default)

Specifies that the state set by the /SET_STATE qualifier remain in effect until explicitly changed. (By default, the /SET_STATE qualifier is in effect only for the next definable key you press or the next read-terminating character that you type.) Can only be specified with the /SET_STATE qualifier.

/LOG (default)
/NOLOG

Specifies whether a log message is displayed after a key is defined. The message indicates a key definition has been created successfully. The /LOG qualifier causes such messages to be displayed and the /NOLOG qualifier suppresses them.

/SET_STATE=state
/NOSET_STATE (default)

Causes the specified state to be set when the key is pressed. (By default, the current locked state is reset when the key is pressed.) A state name can include letters, digits, dollar signs, and underscores. If you omit or negate this qualifier, the current state remains unchanged. You cannot define a key specifying both /SET_STATE and /TERMINATE.

/TERMINATE
/NOTERMINATE (default)

Determines whether the specified command string executes when you press the key. When you use /NOTERMINATE, you must press RETURN to execute the command string. When you use /TERMINATE, VDE executes the command string immediately when you press the key. You cannot define a key specifying both /SET_STATE and /TERMINATE.

EXAMPLES

1
```
VDE> DEFINE/KEY KP1 "SHOW DEFAULTS"
```

This example shows how to define the keypad key KP1 as the VDE command SHOW DEFAULTS. To enter the SHOW DEFAULTS command, press KP1 followed by the RETURN key.

2
```
VDE> DEFINE/KEY KP6 "SET CONTEXT BUGFIX" /TERMINATE
```

This example shows how to define the keypad key 6 as the SET CONTEXT BUGFIX command. The /TERMINATE qualifier causes the SET CONTEXT BUGFIX command to execute when you press keypad key 6 without having to press RETURN.

3
```
VDE> DEFINE/KEY MINUS "SHOW DEFAULTS" /TERMINATE /NOECHO
```

This example shows how to define the minus key on the keypad as the SHOW DEFAULTS command. The /TERMINATE qualifier causes the SHOW DEFAULTS command to execute when you press the minus key. The /NOECHO qualifier prevents the display of the command line on the screen. You see the output of the command, but not the command line itself.

4
```
VDE> DEFINE/KEY/TERMINATE KP7 "SHOW MODULE"
VDE> DEFINE/KEY/SET_STATE=TEXAS PF3 ""
VDE> DEFINE/KEY/IF_STATE=TEXAS/TERMINATE KP7 "SHOW FACILITY"
```

This example shows how to create two definitions for the KP7 key using the /SET_STATE and /IF_STATE qualifiers. The first command defines KP7 as the SHOW MODULE command and executes the command when the key is pressed. This is the default key definition for KP7. The second command establishes a new state for the key: when you press PF3, the state is set to TEXAS. The third command creates a new definition for KP7: when you press KP7, VDE executes the SHOW FACILITY command if the state has been set to TEXAS. You can thus execute the SHOW MODULE command by pressing KP7 by itself, and you can execute the SHOW FACILITY command by pressing the PF3 key followed by the KP7 key.

5
```
VDE> DEFINE/KEY PF3 "SET " /SET_STATE=ALTERED
VDE> DEFINE/KEY KP7 "CONTEXT BUGFIX" /TERMINATE /IF_STATE=ALTERED
VDE> DEFINE/KEY KP8 "CONTEXT FEATURE_1" /TERMINATE /IF_STATE=ALTERED
VDE> DEFINE/KEY KP9 "VERIFY" /TERMINATE /IF_STATE=ALTERED
```

This example shows how to define four different keys and associate them with a state named ALTERED:

1 The first definition defines the key PF3 as the SET command and associates this key with a state named ALTERED.

2 The second definition defines the keypad key 7 as ‘‘CONTEXT BUGFIX’’ and makes it dependent on a state named ALTERED. When you press PF3 followed by keypad key 7, VDE executes the SET CONTEXT BUGFIX command.

3 The third definition defines keypad key 8 as ‘‘CONTEXT FEATURE_1’’ and also makes it dependent on the ALTERED state. When you press PF3 followed by keypad key 8, VDE executes the SET CONTEXT FEATURE_1 command.

4 The fourth definition defines keypad key 9 as ‘‘VERIFY’’ and also makes it dependent on the ALTERED state. When you press PF3 followed by keypad key 9, VDE executes the SET VERIFY command.

DELETE/COMMAND

Deletes command definitions established with the DEFINE/COMMAND command.

FORMAT

DELETE/COMMAND *symbol-name[,symbol-name...]*

PARAMETER

symbol-name

Specifies the name of a command symbol whose definition is to be deleted. This symbol must have been defined with a previous DEFINE/COMMAND command.

QUALIFIERS

None.

EXAMPLES

1

```
VDE> DELETE/COMMAND FOO, FEE
```

This example shows how to delete the command definitions for symbols FOO and FEE.

DELETE/KEY

Deletes key definitions that have been established by the DEFINE/KEY command. The /KEY qualifier is required and must precede any other qualifiers.

FORMAT

DELETE/KEY *[key-name]*

PARAMETER

key-name

Specifies the name of the key to be deleted. If you specify this parameter, do not use the /ALL qualifier.

QUALIFIERS

/ALL

Deletes all key definitions in the specified state; the default is the current state. (Use the /STATE qualifier to specify states.) If you use the /ALL qualifier, do not specify a key name.

/LOG (default)
/NOLOG

Controls whether messages are displayed indicating that the specified key definitions have been deleted. /LOG causes such messages to be displayed and /NOLOG suppresses them.

/STATE=(state-name[,...])
/NOSTATE (default)

Specifies the name of the state for which the specified key definition is to be deleted. The default state is the current state. If you specify only one state name, you can omit the parentheses. State names can be any appropriate alphanumeric string.

EXAMPLES

1

```
VDE> DEFINE/KEY PF3 "SHOW LIBRARY" /TERMINATE
%VDE-I-DEFKEY, DEFAULT key PF3 has been defined
VDE> PF3
VDE> SHOW LIBRARY
Library LIBNAME in root directory DEV$:[PROJDIR]
VDE> DELETE/KEY PF3
%VDE-I-DELKEY, DEFAULT key PF3 has been deleted
VDE> PF3
VDE>
```

In this example, the DEFINE/KEY command defines the PF3 key on the keypad as SHOW LIBRARY. Later the DELETE/KEY command undefines the PF3 key. When the user presses PF3 thereafter, only the VDE prompt is displayed.

2
```
VDE> DELETE/KEY/ALL
%VDE-I-DELKEY, DEFAULT key PF1 has been deleted
%VDE-I-DELKEY, DEFAULT key PF2 has been deleted
%VDE-I-DELKEY, DEFAULT key PF3 has been deleted
%VDE-I-DELKEY, DEFAULT key PF4 has been deleted
VDE>
```

Keys PF1 through PF4 had been defined for state DEFAULT. In this example, the DELETE/KEY command deletes all key definitions in the current state, that is, state DEFAULT. The log messages show what key definitions have been deleted.

DELETE ARCHITECTURE

Deletes one or more architecture table entries in the VDE database. Each architecture table entry...

Requires the DELARCH privilege.

ARCHITECTURE support is not fully implemented.

FORMAT

DELETE ARCHITECTURE *arch-name [, arch-name...]*

PARAMETER

arch-name

The name of the architecture to be deleted. This name may be up to 39 characters long and must follow VDE name syntax.

DESCRIPTION

Under Construction...

QUALIFIERS

/CONFIRM (default)
/NOCONFIRM

Under Construction...

/LOG (default)
/NOLOG

Under Construction...

/REMARK="string"

Specifies a remark string to be stored for the deleted architecture. The quoted remark string, which can be up to 132 characters long, may contain any explanatory remark about the architecture. that you want stored in the VDE database. Typically the remark describes the function or purpose of the architecture.

EXAMPLES

1
```
VDE> DELETE ARCHITECTURE VAXPDP/REMARK="Version n"
VDE>
```

Under Construction...

DELETE BUILD_JOB

Deletes one or more build jobs from the current VDE database.

Requires BUILD privilege.

FORMAT

DELETE BUILD_JOB *stream-name [, stream-name...]*

PARAMETER

stream-name

The name of a development stream. One or more build jobs in this stream are deleted from the database. Use the /IDENTIFICATION qualifier to specify the build jobs to delete. If you do not specify this qualifier, the most recent build job in the stream is deleted.

You can delete build jobs for more than one stream by using wildcard characters in the stream name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Build jobs are deleted for those streams whose names match the wildcard pattern.

DESCRIPTION

The DELETE BUILD_JOB command deletes each build job that matches the specified stream name and build job identification numbers. When a build job is deleted, all information about that build job is deleted from the database of the current VDE library. You cannot delete a build job that is queued for execution or that is currently running. To delete such a build job, stop it using the STOP BUILD_JOB command before you delete it.

QUALIFIERS

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each build job deleted. The /CONFIRM qualifier causes VDE to print a message for each build job asking whether you want that build job deleted. If you answer YES (or Y), the build job is deleted from the database. If you answer NO (or N), the build job is not deleted. The /NOCONFIRM qualifier causes VDE to delete the specified build jobs without asking for confirmation.

/IDENTIFICATION=id-range

Specifies the job identification numbers of the build jobs to be deleted from the specified stream. The **id-range** parameter can be a single integer value, a range of integers, or a wildcard character. If the parameter is a single integer (for example, /IDENT=5), VDE deletes the build job with that identification number. If the parameter is a range of integers (specified as two build job identification numbers separated by a colon as in /IDENT=3:5), VDE deletes all build jobs with identification numbers in that range. Finally, if the **id-range** parameter is the asterisk wildcard (as in /IDENT=*), VDE deletes all build jobs for the specified stream.

If you specify a zero or negative build job identification number in the **id-range** parameter, that number is relative to the identification number of the most recent build job in the stream. For example, if the most recent build job is job 5, then /IDENT=0 is equivalent to /IDENT=5 and /IDENT=–1 is equivalent to /IDENT=4.

If you do not use the /IDENTIFICATION qualifier, VDE deletes the most recent build job for the specified stream.

/LOG (default)
/NOLOG

Controls whether log messages are printed after each build job is deleted. The /LOG qualifier causes such messages to be printed and the /NOLOG qualifier suppresses them. The messages indicate that the build job has been deleted from the database and that the database transaction has successfully committed.

/REMARK="string"

Specifies a deletion remark string to be stored in the database for the deleted build job. The quoted remark string, which can be up to 132 characters long, may contain any explanatory remark that you want to store in the database.

EXAMPLES

1

```
VDE> DELETE BUILD_JOB V2.0-3/IDENTIFICATION=7
%VDE-I-BLDJOBDEL, build job 7 for stream V2.0-3 deleted from database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

In this example, build job 7 for stream V2.0-3 is deleted from the VDE database. The log messages confirm that the build job was deleted and that the database transaction successfully completed.

DELETE CONTEXT

Deletes one or more contexts from the current VDE library. Each context can be used to specify the default development stream, default architecture and default facility for a VDE session.

Requires CRECTX privilege to delete public contexts.

FORMAT

DELETE CONTEXT *ctx-name [, ctx-name...]*

PARAMETER

ctx-name

The name of a context to be deleted from the current VDE library. You can delete more than one context at a time by using wildcard characters in the context name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Those contexts whose names match the wildcard pattern are deleted.

DESCRIPTION

The DELETE CONTEXT command deletes each specified context from the current VDE database. The command deletes either public contexts or your private contexts, depending on whether you specify the /PUBLIC or the /PRIVATE qualifier.

QUALIFIERS

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each context deleted. The /CONFIRM qualifier causes VDE to print a message for each context asking whether you want that context deleted. If you answer YES (or Y), that context is deleted from the database. If you answer NO (or N), the context is not deleted. If you enter a context name with a wildcard specification, VDE asks you to confirm the deletion of each context whose name matches the wildcard specification. The /NOCONFIRM qualifier causes VDE to delete the specified contexts without asking for confirmation.

/LOG (default)
/NOLOG

Controls whether log messages are printed after each context has been deleted. The /LOG qualifier causes such messages to be printed and the /NOLOG qualifier suppresses them. These messages indicate that the context has been deleted from the database and that the database transaction has successfully committed.

/PRIVATE (default)
/PUBLIC

Controls whether a private or public context is deleted. The /PRIVATE qualifier causes VDE to delete a context that is private to you. The

/PUBLIC qualifier causes VDE to delete a context that is available to all users of the library. You must have the CRECTX privilege to use the /PUBLIC qualifier.

/REMARK="string"

Specifies a deletion remark string to be stored in the database for the deleted context. The quoted remark string, which can be up to 132 characters long, may contain any explanatory remark that you want stored in the database.

EXAMPLES

1

```
VDE> DELETE CONTEXT V2RMS
%VDE-I-CTXDEL, context V2RMS deleted from the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example deletes the context V2RMS from the database. Because the /PUBLIC qualifier is not specified, VDE deletes the private context by this name.

DELETE FACILITY

Deletes one or more delete-enabled facilities from the current VDE library.

Requires DELFAC privilege.

FORMAT

DELETE FACILITY *facil-name [, facil-name...]*

PARAMETER

facil-name

The name of a facility to be deleted from the current VDE library. You can delete more than one facility at a time by using wildcard characters in the facility name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Those facilities whose names match the wildcard pattern are deleted.

DESCRIPTION

The DELETE FACILITY command deletes each specified facility from the current VDE database and commits the transaction. It then deletes all disk directories associated with each facility. When the deletion command completes, the files and CMS library associated with the facility will have been vaporized. Should protection violations or other errors occur, VDE can be prevented from deleting all associated files and directories, and VDE will generate one or more informational messages that will provide additional details.

You can only delete a facility if the **delete** attribute is enabled on the facility. If this attribute is disabled, you must enable this attribute, using the MODIFY FACILITY /DELETE command, before you will be able to delete the facility.

QUALIFIERS

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each facility deleted. The /CONFIRM qualifier causes VDE to print a message for each facility asking whether you want that facility deleted. If you answer YES (or Y), that facility is deleted from the library. If you answer NO (or N), the facility is not deleted. If you enter a facility name with a wildcard specification, VDE asks you to confirm the deletion of each facility whose name matches the wildcard specification. The /NOCONFIRM qualifier causes VDE to delete the specified facilities without asking for confirmation.

/LOG (default)
/NOLOG

Controls whether log messages are printed after each facility has been deleted. The /LOG qualifier causes such messages to be printed and the /NOLOG qualifier suppresses them. These messages indicate that

the facility has been deleted from the library and that the database transaction has successfully committed.

/REMARK="string"

Specifies a remark string to be stored in the database for the deleted facility. The quoted remark string, which can be up to 132 characters long, may contain any explanatory remark that you want to store in the database.

EXAMPLES

1

```
VDE> DELETE FACILITY ACCTNG
%VDE-I-FACDEL, facility ACCTNG deleted from the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

In this example, facility ACCTNG is deleted from the current VDE library. The log messages confirm that the facility was deleted and that the database transaction successfully completed.

2

```
VDE> DELETE FACILITY FAC1, FAC2, FAC3
%VDE-I-FACDEL, facility FAC1 deleted from the database
%VDE-I-FACDEL, facility FAC2 deleted from the database
%VDE-I-FACDEL, facility FAC3 deleted from the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example shows how you can delete multiple facilities from a library. The log messages list the facilities that are deleted.

DELETE FOLD

Deletes one or more fold records from the VDE database. A fold record is a reminder that a source change should be manually "folded" into a given development stream in order to propagate the change.

Requires RESREP privilege.

FORMAT

DELETE FOLD *mod-name [, mod-name...]*

PARAMETER

mod-name

Specifies a module for which a fold record is to be deleted. The module name consists of an optional facility name enclosed in square brackets, a module name, and an optional type name preceded by a period (such as [FACIL]MOD1.MAR). If the facility name is omitted, the module is assumed to belong to the default facility. If the type name is omitted, VDE deletes fold records for all source modules with the specified module name in the given facility.

You can delete fold records for more than one module by using wildcard characters in any of the three components of the module name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. VDE deletes fold records for all source modules whose names match the wildcard pattern.

DESCRIPTION

A fold record is a reminder that a source change should be manually folded into a given development stream in order to propagate the change into that stream. The DELETE FOLD command deletes each specified fold record from the VDE database. This command is a synonym for the CANCEL FOLD command.

QUALIFIERS

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each fold record deleted. The /CONFIRM qualifier causes VDE to print a message for each fold record asking whether you want that record deleted. If you answer YES (or Y), that fold record is deleted from the database. If you answer NO (or N), the fold record is not deleted. If you enter a module name with a wildcard specification, VDE asks you to confirm the deletion of fold records for each module whose name matches the wildcard specification. The /NOCONFIRM qualifier causes VDE to delete the specified fold records without asking for confirmation.

/IDENTIFICATION=fold-ident

Specifies the fold record to be deleted. This qualifier is required when you have multiple fold records for the same module in the same stream. The **fold-ident** parameter is the fold identifier of the fold record to be deleted.

The fold identifier is the identifier you specified or VDE assigned when you created the fold record, usually a small integer value. Use the SHOW FOLD command to determine the fold identifier of each fold record.

/LOG (default)
/NOLOG

Controls whether log messages are printed after each fold record has been deleted. The /LOG qualifier causes such messages to be printed and the /NOLOG qualifier suppresses them. These messages indicate that the fold record has been deleted and that the database transaction has successfully committed.

/STREAM=stream-name

Specifies the development stream for which fold records should be deleted. The **stream-name** parameter gives the name of the stream. If this qualifier is omitted, VDE deletes fold records for the default development stream only.

/USERNAME=username

Specifies that fold records are to be deleted on behalf of another user. The **username** parameter specifies the OpenVMS username of the other user. You must have the USERNAME privilege to use this qualifier.

EXAMPLES

1

```
VDE> DELETE FOLD [CODE]X.FOR/IDENT=1
%VDE-I-FOLDDEL, fold record 1 for module [CODE]X.FOR has been deleted
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example deletes fold record 1 for module [CODE]X.FOR. The log messages confirm that the fold record has been successfully deleted.

DELETE GENERATION

Deletes the most recent generation of a specified module from a specified development stream.

Requires PERFREP privilege.

FORMAT

DELETE GENERATION *mod-name [, mod-name...]*

PARAMETER

mod-name

The name of a module whose latest generation in the specified development stream is to be deleted. The module name consists of an optional facility name enclosed in square brackets, a module name, and an optional type name preceded by a period, such as [FACIL]MOD1.MAR. If no facility name is specified, the default facility is assumed. If no type name is specified, the latest generations of all source modules of the specified module name in the given facility are deleted.

You can delete the latest generations of more than one module by using wildcard characters in any of the three components of the module name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. VDE deletes the latest generation in the specified stream of each module whose name matches the wildcard pattern.

Using the /GENERATION qualifier, you can also delete a generation that does not belong to any stream.

DESCRIPTION

The DELETE GENERATION command removes the latest generation of a specified module from a specified development stream and makes the previous generation (the next most recent generation) into the latest generation for the stream. The result is that the most recent generation is discarded from the stream, so that the effect of the REPLACE operation that created that generation is undone.

The removed generation remains in the VDE database as long as it remains reachable from at least one development stream or has at least one successor generation. However, once a removed generation becomes unreachable from all streams and has no successors, the DELETE GENERATION command deletes it completely from the VDE database and library.

You cannot delete the first generation of a module (generation 1) using this command. The only way to delete the first generation is to delete the whole module with the DELETE MODULE command.

If the specified module is currently reserved in the specified stream, VDE terminates that reservation since it is based on a module generation that no longer is the latest generation for the stream. VDE sends a mail message to notify the owner of the reservation that the reservation is lost.

If you want to delete a module generation that does not belong to any development stream, use the /GENERATION qualifier. Do not use the /GENERATION qualifier for a generation that belongs to a stream; VDE will not delete any generation that belongs to a stream or that is not a leaf node in the generation tree.

QUALIFIERS

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each generation deleted. The /CONFIRM qualifier causes VDE to print a message for the latest generation of each module giving the name of the generation and asking whether you want to delete the generation. If you answer YES (or Y), the generation is deleted. If you answer NO (or N), the generation is not deleted. The /NOCONFIRM qualifier causes VDE to delete each generation without asking for confirmation.

/GENERATION=gen-expr

Specifies that the generation with the CMS generation expression given by the **gen-expr** parameter be deleted. If you specify this qualifier, do not specify the /STREAM qualifier.

/LOG (default)
/NOLOG

Controls whether log messages are printed after generations are deleted. The /LOG qualifier causes such messages to be printed and /NOLOG suppresses them. The messages indicate that the generations have been deleted from the specified development stream and that the database transaction has successfully committed.

/REMARK="string"

Specifies a remark string that explains why you deleted the generation. The quoted remark string can be up to 132 characters long.

/STREAM=stream-name

Specifies that the latest generation in the development stream given by the **stream-name** parameter be deleted. If both this qualifier and the /GENERATION qualifier are omitted, VDE prompts you for the stream name.

EXAMPLES

1

```
VDE> DELETE GENERATION [FACIL]MODNAM.C /STREAM=V2.0
%VDE-I-GENDEL, generation [FACIL]MODNAM.C;5(5) deleted from stream V2.0
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
VDE> SHOW GENERATION [FACIL]MODNAM.C
Generation [FACIL]MODNAM.C;4(4)       by user SMITH        on  9-MAY-1990 19:43
VDE>
```

This example deletes the latest generation of module [FACIL]MODNAM.C from stream V2.0. The log message shows that generation 5 was deleted. The subsequent SHOW GENERATION command shows that generation 4 is now the latest generation of the module in the stream.

DELETE GROUP

Deletes one or more groups from the current VDE database.

Requires CREGRP privilege.

FORMAT

DELETE GROUP *group-name [,group-name...]*

PARAMETER

group-name

The name of a source group or build group defined in the default development stream that is to be deleted from the database.

You can delete more than one group at a time by using wildcard characters in the group name. The percent sign (%) in a group name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Those groups whose names match the wildcard pattern are deleted.

DESCRIPTION

This command deletes the database definition of each source group or build group listed as a parameter. The /SOURCE or /BUILD qualifier determines whether source groups or build groups are deleted. Group definitions are deleted only in the default development stream. This command does not affect the modules that are members of the deleted groups, only the definitions of the groups.

QUALIFIERS

/BUILD
/SOURCE (default)

Specifies whether source groups or build groups are deleted. The /SOURCE qualifier causes VDE to delete the source groups with the specified names. The /BUILD qualifier causes VDE to delete the build groups with the specified names. If neither qualifier is specified, VDE deletes source groups.

Of /BUILD and /SOURCE, /SOURCE is the default.

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each group deleted. The /CONFIRM qualifier causes VDE to print a message giving the name of the group and asking if you want it deleted. If you answer YES (or Y), the group definition is deleted from the database. If you answer NO (or N), the group is not deleted. If you use a group specification that contains wildcard characters, VDE asks you to confirm the deletion of each group that matches the wildcard specification. The /NOCONFIRM qualifier causes VDE to delete all specified groups without asking for confirmation.

/LOG (default)
/NOLOG

Controls whether log messages are printed after each group has been deleted. The /LOG qualifier causes such messages to be printed and the /NOLOG qualifier suppresses them. These messages indicate that the group has been deleted and that the database transaction has successfully committed.

/REMARK="string"

Specifies a remark string to be stored in the database for the deleted group. The quoted remark string, which can be up to 132 characters long, may contain any explanatory remark about the group or its deletion that you want to store in the database.

EXAMPLES

1
```
VDE> DELETE GROUP GRPNAM
%VDE-I-GRPDEL, group GRPNAM deleted from the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

In this example, the source group called GRPNAM is deleted from the VDE database in the default stream. The log messages confirm that the group was deleted and that the corresponding database transaction completed successfully.

2
```
VDE> DELETE GROUP/BUILD GRP1, GRP2, GRP33
%VDE-I-GRPDEL, group GRP1 deleted from the database
%VDE-I-GRPDEL, group GRP2 deleted from the database
%VDE-I-GRPDEL, group GRP33 deleted from the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example shows how to specify a list of group names with the DELETE GROUP command. The log messages list the groups that were deleted. Because the /BUILD qualifier is specified, this command only deletes build groups.

3
```
VDE> DELETE GROUP/CONFIRM GROUP_1, GROUP_2
Delete group GROUP_1 ? [No]: Y
%VDE-I-GRPDEL, group GROUP_1 deleted from the database
Delete group GROUP_2 ? [No]: N
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example illustrates the /CONFIRM qualifier. This qualifier causes VDE to ask the user whether each specified group should be deleted. For group GROUP_1, the user answers Y (for YES) and that group is deleted from the database, as the log message shows. For group GROUP_2, the user answers N (for NO), that group is not deleted, and no log message is displayed. The final log message indicates that the database transaction for the whole command has successfully committed.

4
```
VDE> DELETE GROUP GRP*
%VDE-I-GRPDEL, group GRP1 deleted from the database
%VDE-I-GRPDEL, group GRP2 deleted from the database
%VDE-I-GRPDEL, group GRP22 deleted from the database
%VDE-I-GRPDEL, group GRP33 deleted from the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

In this example, the group specification (GRP*) contains a wildcard character. The asterisk matches zero or more characters. Source groups GRP1, GRP2, GRP22, and GRP33 match the group specification and are deleted as the log messages show.

5
```
VDE> DELETE GROUP GRP%
%VDE-I-GRPDEL, group GRP1 deleted from the database
%VDE-I-GRPDEL, group GRP2 deleted from the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example is identical to the previous example except that the percent sign is used as a wildcard character. The percent sign matches exactly one character. As a result, source groups GRP1 and GRP2 (but not GRP22 or GRP33) match the group specification (GRP%) and are deleted.

DELETE MODULE

Deletes one or more modules from the current VDE library.

Requires DELMOD privilege or, in some cases, CREMOD privilege.

FORMAT

DELETE MODULE *mod-name [, mod-name...]*

PARAMETER

mod-name

The name of a module to be deleted from the current VDE library. This name consists of a module name and a type name separated by a period (such as MOD1.PAS) and can be preceded by an optional facility name enclosed in square brackets (as in [FACIL]MOD1.PAS). If no facility name is specified, the current default facility is assumed.

You can delete more than one module at a time by using wildcard characters in any of the three components of the module name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Those modules whose names match the wildcard pattern are deleted.

DESCRIPTION

The DELETE MODULE command deletes each specified module from the current VDE library. When a module is deleted, VDE deletes its database record and removes the module from the build dependency graph and from any group definitions. After committing the database transaction, VDE also deletes each specified module's delta file (CMS element). However, VDE does not remove the module from any stream-specific disk directories such as clear-copy areas or object directories.

Normally, you must have the DELMOD privilege to delete a module. However, if you have created a particular module and you have not yet created the first generation of the module (normally done by replacing the first generation into the VDE library), you are allowed to delete the module without the DELMOD privilege. In this particular case, you must have the CREMOD privilege, which was required for the initial module creation. The creator of a module can thus correct simple mistakes, such as misnaming the module, by simply deleting the module before the initial replacement is performed. The outstanding reservation on the module is canceled, the module is removed from any reservation session that it might be included in, and the module never appears in the VDE library. For further information on the CREATE MODULE command, see CREATE MODULE.

If you are not the creator of a module or if the module already has at least one generation, you must have the DELMOD privilege to delete it and you can only delete a module if the **delete** attribute is set for the module. If this attribute is not set, you must set it with the MODIFY MODULE /DELETE command before you can delete the module.

DELETE MODULE

QUALIFIERS

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each module deleted. The /CONFIRM qualifier causes VDE to print a message for each module asking whether you want that module deleted. If you answer YES (or Y), that module is deleted from the library. If you answer NO (or N), the module is not deleted. If you enter a module name with a wildcard specification, VDE asks you to confirm the deletion of each module that matches the wildcard specification. The /NOCONFIRM qualifier causes VDE to delete the specified modules without asking for confirmation.

/LOG (default)
/NOLOG

Controls whether log messages are printed after each module has been deleted. The /LOG qualifier causes such messages to be printed and the /NOLOG qualifier suppresses them. These messages indicate that the module has been deleted from the library and that the database transaction has successfully committed.

/REMARK="string"

Specifies a remark string to be stored in the database for the deleted module. The quoted remark string, which can be up to 132 characters long, may contain any explanatory remark that you want to store in the database.

EXAMPLES

1
```
VDE> DELETE MODULE MOD1.PAS
%VDE-I-MODDEL, module [CODE]MOD1.PAS deleted from the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

In this example, module MOD1.PAS is deleted from the current VDE library. Since no facility name is specified, the module is assumed to belong to the current default facility, facility CODE. The log messages confirm that the module was deleted and that the database transaction successfully completed.

2
```
VDE> DELETE MODULE [FACIL]FOO.MAR, FOO.OBJ, [RMS]MOD2.FOR
%VDE-I-MODDEL, module [FACIL]FOO.MAR deleted from the database
%VDE-I-MODDEL, module [FACIL]FOO.OBJ deleted from the database
%VDE-I-MODDEL, module [RMS]MOD2.FOR deleted from the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example shows how to delete multiple modules from the library. The log messages list the deleted modules.

DELETE REPLACEMENT

Deletes one or more queued replacements from the VDE database to free up database space. Most users should use CANCEL REPLACEMENT instead.

Requires DELREP privilege.

FORMAT

DELETE REPLACEMENT *rep-name [, rep-name...]*

PARAMETER

rep-name

The name of a queued replacement to be deleted from the current VDE library. You can delete more than one replacement at a time by using wildcard characters in the replacement name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Those queued replacements whose names match the wildcard pattern are deleted.

Do not specify the **rep-name** parameter if you use the /IDENTIFICATION qualifier.

DESCRIPTION

The DELETE REPLACEMENT command deletes each specified queued replacement from the current VDE database. This command is used to remove replacement records from the database to free up database space. It can be used to delete both current replacement records and historical replacement data. For this reason, it requires the DELREP privilege.

Users who simply want to cancel a current replacement should use the CANCEL REPLACEMENT command. This command requires the RESREP privilege to cancel your own replacement or the PERFREP privilege to cancel another user's replacement.

QUALIFIERS

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each replacement deleted. The /CONFIRM qualifier causes VDE to print a message for each replacement asking whether you want that replacement deleted. If you answer YES (or Y), that replacement is deleted from the database. If you answer NO (or N), the replacement is not deleted. If you enter a replacement name with a wildcard specification, VDE asks you to confirm the deletion of each replacement whose name matches the wildcard specification. The /NOCONFIRM qualifier causes VDE to delete the specified replacements without asking for confirmation.

/HISTORY

Specifies that replacement history be deleted. Replacement history consists of database records for past replacements that have already been performed; it does not include currently queued replacements. When used

with the /IDENTIFICATION qualifier, the /HISTORY qualifier deletes information for chronological ranges of past replacements.

If you omit the /HISTORY qualifier, VDE deletes currently queued replacements only.

/IDENTIFICATION=id-range

Specifies that the replacements with the unique Replacement Ids given by the **id-range** parameter be deleted. Do not specify the **rep-name** parameter when you use this qualifier.

VDE assigns a unique Replacement Id to each immediate or queued replacement into the VDE library. VDE assigns Replacement Ids in chronological order so that the first replacement into the library has Id value 1, the next has Id value 2, and so on. When you use this qualifier, all replacements that have Id values in the range given by the **id-range** parameter are deleted.

The **id-range** parameter can be a single integer value, a range of integers, or a wildcard character. If the parameter is a single integer (for example, /IDENT=5), VDE deletes the replacement with that Replacement Id. If the parameter is a range of integers (specified as two integers separated by a colon as in /IDENT=3:5), VDE deletes the replacements with Ids in that range. Finally, if the **id-range** parameter is the asterisk wildcard character (as in /IDENT=*), VDE deletes all replacements.

If you specify a zero or negative number in the **id-range** parameter, that number is relative to the largest Replacement Id value for the VDE library (the latest replacement). For example, if the latest replacement has Replacement Id 250, then /IDENT=0 is equivalent to /IDENT=250 and /IDENT=–1 is equivalent to /IDENT=249.

The /IDENTIFICATION qualifier can be used with the /HISTORY qualifier to delete historical replacement information from the VDE database. Deleting such information allows database space to be reused.

/LOG (default)
/NOLOG

Controls whether log messages are printed after each replacement has been deleted. The /LOG qualifier causes such messages to be printed and the /NOLOG qualifier suppresses them. These messages indicate that the replacement has been deleted from the database and that the database transaction has successfully committed.

EXAMPLES

1
```
VDE> DELETE REPLACEMENT ANDERSON-1
%VDE-I-REPDEL, replacement ANDERSON-1 deleted from the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

In this example, replacement ANDERSON–1 is deleted from the VDE database. The log messages confirm that the replacement was deleted and that the database transaction successfully completed.

DELETE SCRIPT

Deletes a VDE script from the VDE database.

Requires CRESCRP privilege.

FORMAT DELETE SCRIPT

DESCRIPTION

The DELETE SCRIPT command deletes a specified VDE script from the VDE database. VDE only deletes the script of the specified type and segment number that belongs to the default development stream.

You must specify a script-type qualifier to select the script you want deleted. See the description of the CREATE SCRIPT command for an explanation of the different types of scripts and how scripts are selected using script-type qualifiers. You must also specify the script segment number unless the default value applies.

Keywords used within scripts are documented in Script Keywords, while script functions are in Script Functions, and script types are in Script Types. For other script-related commands, also see CREATE SCRIPT, EXTRACT SCRIPT, INVOKE SCRIPT, MODIFY SCRIPT, and SHOW SCRIPT.

QUALIFIERS

/BUILD

Specifies that the default stream's BUILD script be deleted.

/COMPILE=mod-name

Specifies that the default stream's COMPILE script with the specified **mod-name** specification be deleted.

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each script deleted. The /CONFIRM qualifier causes VDE to print a message for each script asking whether you want that script deleted. If you answer YES (or Y), the script is deleted from the database. If you answer NO (or N), the script is not deleted. The /NOCONFIRM qualifier causes VDE to delete the specified scripts without asking for confirmation.

/COPY=mod-name

Specifies that the default stream's COPY script with the specified **mod-name** specification be deleted.

/CREATE_FACILITY

Specifies that the script that VDE invokes when a new facility is created should be deleted.

/DELETE_GENERATION=mod-name
Specifies that the default stream's DELETE_GENERATION script with the specified **mod-name** specification be deleted.

/DIFFERENCES=mod-name
Specifies that the default stream's DIFFERENCES script with the specified **mod-name** specification be deleted.

/FETCH=mod-name
Specifies that the default stream's FETCH script with the specified **mod-name** specification be deleted.

/INSERT_GENERATION=mod-name
Specifies that the default stream's INSERT_GENERATION script with the specified **mod-name** specification be deleted.

/LINK=mod-name
Specifies that the default stream's LINK script with the specified **mod-name** specification be deleted.

/LOG (default)
/NOLOG
Controls whether log messages are printed after the script is deleted. The /LOG qualifier causes such messages to be printed and the /NOLOG qualifier suppresses them. These messages indicate that the script has been deleted and that the database transaction has committed successfully.

/NEW_STREAM=fac-name
Specifies that the default stream's NEW_STREAM script with the specified **fac-name** specification be deleted.

/NOKEEP=mod-name
Specifies that the default stream's NOKEEP script with the specified **mod-name** specification be deleted.

/REPLACE=mod-name
Specifies that the default stream's REPLACE script with the specified **mod-name** specification be deleted.

/RESERVE=mod-name
Specifies that the default stream's RESERVE script with the specified **mod-name** specification be deleted.

/SEGMENT=seg-num
Specifies the segment number of the script to delete. The **seg-num** parameter specifies the integer segment number. If you omit this qualifier, VDE deletes the script with segment number 10.

/SET_FACILITY=fac-name
Specifies that the default stream's SET_FACILITY script with the specified **fac-name** specification be deleted.

/SET_STREAM
Specifies that the default stream's SET_STREAM script be deleted.

/STAGING=mod-name

Specifies that the default stream's STAGING script with the specified **mod-name** specification be deleted.

/UNRESERVE=mod-name

Specifies that the default stream's UNRESERVE script with the specified **mod-name** specification be deleted.

EXAMPLES

1

```
VDE> DELETE SCRIPT/COMPILE=[CODE]*.C
%VDE-I-SCRPDEL, script deleted from the database
 script to compile module [CODE]*.C (segment 10)
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example deletes the script to compile all C modules in facility CODE. The log messages identify the script and show that it was successfully deleted. The script is only deleted for the default development stream.

DELETE STEP

Deletes build-step dependency information from the VDE database. This information specifies the input and output modules of a given compile, copy, or link step.

Requires CRESTEP privilege.

FORMAT

DELETE STEP *mod-name [, mod-name...]*

PARAMETER

mod-name

The name of the module that is compiled, copied, or linked by the build step to be deleted from the database. This name consists of an optional facility name enclosed in square brackets, a module name, and an optional type name preceded by a period (for example, [FACNAM]MOD1.PAS). If no facility name is specified, the current default facility is assumed. If no type name is specified, all build steps for modules with the given facility name and module name are deleted.

You can delete more than one build step at a time by using wildcard characters in any of the three components of the module name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Those build steps whose module names match the wildcard pattern are deleted.

DESCRIPTION

The DELETE STEP command deletes all dependency information for a given build step so that the database no longer records the input and output modules for the step. The modules themselves remain in the database; only their relationships to the build step are lost.

The command deletes build-step dependency information from only the default development stream. To delete build steps in more than one stream, you must repeat the DELETE STEP command for each stream. For each stream, first enter the SET STREAM command to set the development stream, then enter the DELETE STEP command to delete the desired build-step dependencies from the database.

QUALIFIERS

/COMPILE (default)
/COPY
/LINK

Specifies the kind of build step to be deleted. /COMPILE deletes the compile step, /COPY deletes the copy step, and /LINK deletes the link step for the specified module.

Of /COMPILE, /COPY and /LINK, /COMPILE is the default.

/CONFIRM (default)
/NOCONFIRM

Controls whether VDE asks you to confirm that you want the build-step definition deleted. The /CONFIRM qualifier causes VDE to print a message for each build step asking whether you want that step deleted. If you answer YES (or Y), the build-step definition is deleted. If you answer NO (or N), the build-step definition is not deleted. The /NOCONFIRM qualifier causes VDE to delete all specified build-step definitions without asking for confirmation.

/LOG (default)
/NOLOG

Controls whether log messages are printed when build-step definitions are deleted from the database. The /LOG qualifier causes the messages to be printed and the /NOLOG qualifier suppresses them. The messages indicate that each build step has been deleted and that the database transaction has committed successfully.

EXAMPLES

1
```
VDE> DELETE STEP [RMS]FOO.MAR
%VDE-I-COMSTEPDEL, compile step for module [RMS]FOO.MAR deleted from database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example deletes all build dependency information for the compilation of module FOO.MAR in facility RMS.

2
```
VDE> DELETE STEP/LINK FOO.EXE, MOD1.EXE
%VDE-I-LNKSTEPDEL, link step for module [CODE]FOO.EXE deleted from database
%VDE-I-LNKSTEPDEL, link step for module [CODE]MOD1.EXE deleted from database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This command deletes the dependency information for the link steps for modules FOO.EXE and MOD1.EXE in the current default facility, facility CODE.

DELETE STREAM

Deletes one or more development streams from the current VDE library.

Requires DELSTRM privilege.

FORMAT

DELETE STREAM *stream-name [, stream-name...]*

PARAMETER

stream-name

The name of a stream to be deleted from the current VDE library. You can delete more than one stream at a time by using wildcard characters. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Those streams whose names match the wildcard pattern are deleted.

DESCRIPTION

The DELETE STREAM command deletes each specified development stream from the current VDE library. When a stream is deleted, all database information associated with the stream is deleted. The deleted database information includes the stream's group definitions, scripts, and build dependency graph. The stream is also removed from any stream successor chains it belongs to. The stream's disk directories remain unchanged, however; they are not deleted.

You can only delete a stream if the **delete** attribute is set for the stream. If this attribute is not set, you must set it with the MODIFY STREAM /DELETE command before you can delete the stream. Unless the library's allow-deletion attribute is set, the stream must also be closed before it can be deleted. (Use MODIFY STREAM /CLOSED to close a stream.)

QUALIFIERS

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each stream deleted. The /CONFIRM qualifier causes VDE to print a message for each stream asking whether you want that stream deleted. If you answer YES (or Y), that stream is deleted from the library. If you answer NO (or N), the stream is not deleted. If you enter a stream name with a wildcard specification, VDE asks you to confirm the deletion of each stream whose name matches the wildcard specification. The /NOCONFIRM qualifier causes VDE to delete the specified streams without asking for confirmation.

/LOG (default)
/NOLOG

Controls whether log messages are printed after each stream has been deleted. The /LOG qualifier causes such messages to be printed and the /NOLOG qualifier suppresses them. These messages indicate that

the stream has been deleted from the library and that the database transaction has successfully committed.

/REMARK="string"

Specifies a remark string to be stored in the database for the deleted stream. The quoted remark string, which can be up to 132 characters long, may contain any explanatory remark that you want to store in the database.

EXAMPLES

1

```
VDE> DELETE STREAM V3.1-1
%VDE-I-STRMDEL, stream V3.1-1 deleted from the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

In this example, development stream V3.1-1 is deleted from the current VDE library. The log messages confirm that the stream was deleted and that the database transaction successfully completed.

2

```
VDE> DELETE STREAM V2.0-BL1, V2.0-BL2, V2.0-BL3
%VDE-I-STRMDEL, stream V2.0-BL1 deleted from the database
%VDE-I-STRMDEL, stream V2.0-BL2 deleted from the database
%VDE-I-STRMDEL, stream V2.0-BL3 deleted from the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example shows how to specify a list of stream names to delete multiple streams from the library. The log messages list the deleted streams.

DELETE USER

Deletes user records from the database in the current VDE library. This command deletes the database record for each user whose OpenVMS username is listed as a parameter and denies that user access to the current VDE library.

Requires CREUSR privilege.

FORMAT

DELETE USER *username [, username...]*

PARAMETER

username

The OpenVMS username of the VDE user whose record is deleted from the database of the current VDE library. You can delete more than one username at a time by using wildcard characters. The percent sign (%) in a username matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Those users whose usernames match the wildcard pattern are deleted.

DESCRIPTION

The DELETE USER command disables access by, and disallows any command specification of, the specified user. It also cancels all review and notification entries for the specified user.

The command marks the specified user records in the database as deleted, though the user records are not physically deleted from the database. As a result of this behaviour, records in the VDE database can continue to refer to the user records marked as deleted. For example, the record for each module indicates who created the module, and the SHOW MODULE command shows the creator's username and full name string. That information is found in the creator's user record, even if the creator has since left the project and a DELETE USER command has marked the associated user record as deleted.

A user that lacks a matching user entry, or has a deleted user entry, will be unable to access the VDE utility[1] Further, usernames that have been deleted will *not* be accepted on VDE commands. However, the username record does remain within the database, should manual recovery be required at a later date.

Lists of valid VDE users can also be automatically maintained using the VDE/READ_UAF command. VDE/READ_UAF compares the current list of users stored in the VDE database with those stored in the OpenVMS system authorization database (SYSUAF). The VDE/READ_UAF command is described in VDE/READ_UAF.

[1] This assumes that the VDE library option that allows automatic addition of new users upon first access is disabled.

As an alternative to the DELETE USER command, one could clear all bits in the associated privilege mask, and/or one could use the MODIFY USER/USER command to rename the user. These steps prevent the user from performing operations on the VDE database, but these options preserve the ability to specify the username in VDE commands.

QUALIFIERS

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each user record deleted from the database. The /CONFIRM qualifier causes VDE to print a message for each user record giving the username and asking whether you want that record deleted. If you answer YES (or Y), that user record is deleted from the database. If you answer NO (or N), the user record is not deleted. If you enter a username with a wildcard specification, VDE asks you to confirm the deletion of each username that matches the wildcard specification. The /NOCONFIRM qualifier causes VDE to delete the specified users from the database without asking for confirmation.

/LOG (default)
/NOLOG

Controls whether log messages are printed after each user record has been deleted. These messages indicate that the user has been deleted from the database and that the database transaction has successfully committed. The /LOG qualifier causes such messages to be printed and the /NOLOG qualifier suppresses them.

/REMARK="string"

Specifies a remark string to be stored in the database for the deleted user record. The quoted remark string, which can be up to 132 characters long, may contain any explanatory remark that you want to store in the database.

EXAMPLES

1

```
VDE> DELETE USER SMITH
%VDE-I-USERDEL, user SMITH deleted from the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

In this example, the record for user SMITH is deleted from the database of the current VDE library. The log messages confirm that the user record was deleted and that the database transaction successfully completed.

2

```
VDE> DELETE USER SMITH, DOE, JONES
%VDE-I-USERDEL, user SMITH deleted from the database
%VDE-I-USERDEL, user DOE deleted from the database
%VDE-I-USERDEL, user JONES deleted from the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example shows how to delete multiple users from the database. The log messages list the users who have been deleted.

3
```
VDE> DELETE USER/CONFIRM SMITH, DOE
Delete user SMITH ? [No]: Y
%VDE-I-USERDEL, user SMITH deleted from the database
Delete user DOE ? [No]: N
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example illustrates the /CONFIRM qualifier. This qualifier causes VDE to ask the user whether each specified user record should be deleted from the database. For user SMITH, the answer is Y (for YES) and that user record is deleted from the database. For user DOE, the answer is N (for NO) and that user record is not deleted. Hence there is no log message for user DOE. The final log message indicates that the database transaction for the whole command has successfully committed.

4
```
VDE> DELETE USER %O*
%VDE-I-USERDEL, user DOE deleted from the database
%VDE-I-USERDEL, user JONES deleted from the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

In this example, the username specification (%O*) contains wildcard characters. All usernames with an ‘‘O’’ in the second position match the specification. Hence users DOE and JONES are deleted from the database.

DIFFERENCES

Displays the differences between any two module generations.

FORMAT

DIFFERENCES *gen-spec-1 [gen-spec-2]*

PARAMETER

gen-spec-1

Specifies the first module generation to be compared. The generation specification consists of the following in order:

- an optional facility name enclosed in square brackets
- a module name
- a type name
- an optional generation number preceded by a semicolon

An example of a generation specification is [RMS]MOD1.MAR;5.

If you omit the facility name, a module generation from the default facility is compared. If you omit the generation number, the most recent generation of the module in the specified (or default) stream is compared.

If you specify a zero or negative generation number, that number is relative to the most recent generation of the module in the stream specified by the /STREAM qualifier (or the default stream if that qualifier is omitted). For example, if the most recent generation is generation 5, then generation 0 is the same as generation 5 and generation –1 is the same as generation 4.

If you specify the /GENERATION qualifier, VDE compares the generation with the given CMS generation expression and ignores the generation number on the **gen-spec-1** parameter.

gen-spec-2

Specifies the second module generation to be compared. This parameter can have the same form as the first parameter (**gen-spec-1**) to fully specify a VDE generation. If the facility, module, and type names are the same as for the first parameter, this second parameter can be abbreviated to just the generation number, optionally preceded by a semicolon. For example, ‘‘;5’’ or just ‘‘5’’ is enough to specify generation 5 of the module given by the **gen-spec-1** parameter.

If the **gen-spec-2** parameter is omitted, VDE compares the generation given by the **gen-spec-1** parameter with the immediately preceding generation. You can also use the /GENERATION and /STREAM qualifiers to compare other generations of the same module.

DIFFERENCES

DESCRIPTION

The DIFFERENCES command determines the differences between any two module generations in the VDE library. It fetches the two specified generations into a scratch directory and runs the DCL DIFFERENCES command in a subprocess to compare the texts of the two generations and to display the differences between them on your terminal.

The command fetches the specified generations into a scratch directory determined as follows. If logical name VDE$SCRATCH is defined, VDE fetches the files into the directory that this logical name is bound to. If this logical name is not defined, VDE fetches the files into a VDE-created subdirectory [.VDE$SCRATCH] under the current SYS$SCRATCH directory. (By default, SYS$SCRATCH is defined as your SYS$LOGIN login directory. This logical name can be redefined by a user to reference a user-created subdirectory; all scratch files can thus be corraled in a particular subdirectory.) VDE creates the [.VDE$SCRATCH] subdirectory if it does not already exist, and defines the logical name VDE$SCRATCH to reference it. VDE then spawns a subprocess to run the DCL DIFFERENCES command and display the differences. VDE deletes the fetched files at the end of the command unless you specify the /KEEP qualifier. The difference output goes to your terminal by default, but you can direct it to a file using the /OUTPUT qualifier.

Although the DIFFERENCES command can compute the differences between any two generations in the VDE library, it is almost always used to compare two generations of the same module. For this reason, you can reduce the second generation specification parameter so it consists of only a generation number. The following example compares generations 5 and 2 of module [FACIL]MOD.TYP for your default stream:

```
VDE> DIFFERENCES [FACIL]MOD.TYP;5 ;2
```

You can also omit the second generation specification parameter entirely. In this case, the specified generation is compared to its immediate predecessor. The following command thus compares generation 5 with generation 4:

```
VDE> DIFFERENCES [FACIL]MOD.TYP;5
```

When you omit the second parameter, you can use the /GENERATION qualifier to specify the CMS generation expressions of the two generations to compare. For example, the following command compares the generations with CMS generation expressions 10A2 and 11B1:

```
VDE> DIFFERENCES [FACIL]MOD.TYP/GENERATION=(10A2,11B1)
```

You can also use the /STREAM qualifier to specify that the latest generations from two streams be compared. The following example illustrates this use of the /STREAM qualifier:

```
VDE> DIFFERENCES [FACIL]MOD.TYP/STREAM=(V2.0,V2.2)
```

This example compares the latest generations of the given module in streams V2.0 and V2.2.

In addition to accepting certain VDE-specific qualifiers, the DIFFERENCES command accepts all qualifiers that the DCL DIFFERENCES command accepts. These qualifiers are simply passed through to the subprocess that runs the OpenVMS DIFFERENCES program.

QUALIFIERS

/GENERATION=gen-expr

Specifies that the module generation that has the CMS generation expression given by the **gen-expr** parameter be compared. If this qualifier is omitted, VDE uses the generation given by the generation number on the module-name parameter. If both this qualifier and the generation number are omitted, the command uses the latest generation of the module for the specified development stream.

/KEEP

Specifies that one or both of the fetched and compared module generations be retained in the scratch directory after the command completes. If this qualifier is omitted, the files containing those generations are deleted after the differences are computed. If you specify the /KEEP qualifier after the **gen-spec-1** parameter, the file containing that generation is kept. Likewise, if you specify the /KEEP qualifier after the **gen-spec-2** parameter, the file containing that generation is kept. If you specify the qualifier after the DIFFERENCES keyword or after both parameters, the files containing both generations are kept.

/STREAM=stream-name

Specifies that the generation to be compared should be taken from the development stream given by the **stream-name** parameter. If this qualifier is omitted, the generation is taken from your default stream unless you use the /GENERATION qualifier.

/CHANGE_BAR
/COMMENT_DELIMITER
/IGNORE
/MAXIMUM_DIFFERENCES
/MATCH
/MERGED
/MODE
/NUMBER
/NONUMBER
/OUTPUT
/PARALLEL
/SEPARATED
/SLP
/WIDTH
/WINDOW

These qualifiers are passed through to the spawned DCL DIFFERENCES command without change. See the description of the DIFFERENCES

DIFFERENCES

command in the *OpenVMS DCL Dictionary* manual for a description of these qualifiers and their parameters.

EXAMPLE

```
VDE> DIFFERENCES [FACIL]A.B32
************
File DEVD$:[SMITH.VDE$SCRATCH]A.REQ;2
     9   !       declarations.  Here is an additional comment.
    10   !
******
File DEVD$:[SMITH.VDE$SCRATCH]A.REQ;1
     9   !       declarations.
    10   !
************

Number of difference sections found: 1
Number of difference records found: 1

DIFFERENCES /IGNORE=()/MERGED=1-
    DEVD$:[SMITH.VDE$SCRATCH]A.REQ;2-
    DEVD$:[SMITH.VDE$SCRATCH]A.REQ;1
VDE>
```

This example compares the default stream's two most recent generations of module [FACIL]A.B32 and displays their differences on the terminal.

EXIT

Ends the VDE session or ends execution of commands in a command procedure.

FORMAT

EXIT

PARAMETERS

None.

DESCRIPTION

When you enter the EXIT command at the terminal, you end the current VDE session. (Pressing Ctrl/Z also ends a VDE session.) When VDE executes an EXIT command in a command procedure, it returns control to the command stream that invoked the command procedure. That command stream may be the terminal or another command procedure.

QUALIFIERS

/KEPT_PROCESS

If you enter the EXIT/KEPT_PROCESS command at the terminal (not in a VDE command procedure) and the VDE utility is running in a kept subprocess, this qualifier causes VDE to terminate that subprocess before it returns you to DCL level. The advantage of terminating the kept subprocess is that it no longer consumes resources. The disadvantage is that your next VDE command will require more time to reactivate the VDE image and reconnect to the VDE database.

EXAMPLES

1

```
VDE> EXIT
$
```

This example shows how the EXIT command returns control to DCL.

EXTRACT SCRIPT

Extracts a VDE script from the VDE database and writes it to a file.

FORMAT

EXTRACT SCRIPT *file-spec*

PARAMETER

file-spec

Specifies the OpenVMS file specification for the output file that will contains the text of the extracted script. If you omit the directory specification from the **file-spec** parameter, your current default directory is assumed. If you omit the file extension from the **file-spec** parameter, file extension VDESCRIPT is assumed.

DESCRIPTION

The EXTRACT SCRIPT command reads a specified VDE script from the VDE database and writes it to an ASCII text file. VDE only extracts the script of the specified type and segment number that belongs to the default development stream. You can use the EXTRACT SCRIPT command when you want to get a script into a file so that you can edit it and then reinsert it into the database with a subsequent CREATE SCRIPT command.

You must specify a script-type qualifier to select the script you want extracted. See the description of the CREATE SCRIPT command for an explanation of the format of scripts, the different types of scripts, and how scripts are selected using script-type qualifiers. You must also specify the script segment number unless the default value applies.

Keywords used within scripts are documented in Script Keywords, while script functions are in Script Functions, and script types are in Script Types. For other script-related commands, also see CREATE SCRIPT, DELETE SCRIPT, INVOKE SCRIPT, MODIFY SCRIPT, and SHOW SCRIPT.

QUALIFIERS

/BUILD

Specifies that the default stream's BUILD script be extracted.

/COMPILE=mod-name

Specifies that the default stream's COMPILE script with the specified **mod-name** specification be extracted.

/COPY=mod-name

Specifies that the default stream's COPY script with the specified **mod-name** specification be extracted.

/CREATE_FACILITY

Specifies that the script that VDE invokes when a new facility is created should be extracted.

/DELETE_GENERATION=mod-name
Specifies that the default stream's DELETE_GENERATION script with the specified **mod-name** specification be extracted.

/DIFFERENCES=mod-name
Specifies that the default stream's DIFFERENCES script with the specified **mod-name** specification be extracted.

/FETCH=mod-name
Specifies that the default stream's FETCH script with the specified **mod-name** specification be extracted.

/INSERT_GENERATION=mod-name
Specifies that the default stream's INSERT_GENERATION script with the specified **mod-name** specification be extracted.

/LINK=mod-name
Specifies that the default stream's LINK script with the specified **mod-name** specification be extracted.

/LOG (default)
/NOLOG
Controls whether log messages are printed after the script is extracted. The /LOG qualifier causes such messages to be printed and the /NOLOG qualifier suppresses them. These messages indicate that the script has been extracted and identify the output file.

/NEW_STREAM=fac-name
Specifies that the default stream's NEW_STREAM script with the specified **fac-name** specification be extracted.

/NOKEEP=mod-name
Specifies that the default stream's NOKEEP script with the specified **mod-name** specification be extracted.

/REPLACE=mod-name
Specifies that the default stream's REPLACE script with the specified **mod-name** specification be extracted.

/RESERVE=mod-name
Specifies that the default stream's RESERVE script with the specified **mod-name** specification be extracted.

/SEGMENT=seg-num
Specifies the segment number of the script to extract. The **seg-num** parameter specifies the integer segment number. If you omit this qualifier, VDE extracts the script with segment number 10.

/SET_FACILITY=fac-name
Specifies that the default stream's SET_FACILITY script with the specified **fac-name** specification be extracted.

/SET_STREAM
Specifies that the default stream's SET_STREAM script be extracted.

/STAGING=mod-name

Specifies that the default stream's STAGING script with the specified **mod-name** specification be extracted.

/UNRESERVE=mod-name

Specifies that the default stream's UNRESERVE script with the specified **mod-name** specification be extracted.

EXAMPLES

1

```
VDE> EXTRACT SCRIPT/COMPILE=[CODE]*.C [MYDIR]SCRIPT_C
%VDE-I-SCRPEXTR, script extracted to file DEV$:[MYDIR]SCRIPT_C.VDESCRIPT;1
 script to compile module [CODE]*.C (segment 10)
VDE> EXIT
$ TYPE [MYDIR]SCRIPT_C.VDESCRIPT
! Script to compile a C module.
!
EMIT "$ SET VERIFY"
EMIT "$ DEFINE/NOLOG SRC$ ", $DIR(SRC)
EMIT "$ DEFINE/NOLOG OBJ$ ", $DIR(OBJ)
EMIT "$ CC/DEBUG/OBJ=OBJ$:/ANA=OBJ$: SRC$:",$MODTYP
EMIT "$ VDE/DEPEND C OBJ$:",$MOD,".ANA"
EXIT
$
```

This example extracts the script to compile C modules in facility CODE. VDE writes the script into file DEV$:[MYDIR]SCRIPT_C.VDESCRIPT. The subsequent TYPE command shows that the file contains the text of the script.

FETCH

Retrieves a generation of one or more source modules from the current VDE library.

FORMAT

FETCH *gen-spec [, gen-spec...] ["remark"]*

PARAMETER

gen-spec

Specifies a generation of a source module in the default stream to be retrieved from the library. A generation specification consists of the following in order:

- an optional facility name enclosed in square brackets
- a module name
- an optional type name preceded by a period
- an optional generation number preceded by a semicolon

An example of a generation specification is [RMS]MOD1.MAR;5.

If you omit the facility name, modules from the default facility are fetched. If you omit the type name, all source modules with the specified module name in the specified facility are fetched. If you omit the generation number, the most recent generation of the module in the requested development stream is fetched. This is either the stream specified with the /STREAM qualifier or the default development stream.

If you specify a zero or negative generation number, that number is relative to the most recent generation of the module in the requested development stream. For example, if the most recent generation is generation 5, then generation 0 is the same as generation 5 and generation –1 is the same as generation 4.

If you specify the /GENERATION qualifier, VDE fetches the generation with the given CMS generation expression and ignores the generation number on the **gen-spec** parameter.

You can fetch more than one module by using wildcard characters in the facility name, module name, or type name. (Wildcard characters are not allowed in the generation number.) The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Those source modules whose names match the wildcard pattern are fetched.

You can also fetch multiple modules by specifying the name of a source group instead of a generation expression. Source groups are created with the CREATE GROUP command. If you specify a group name, the latest generation of each module that is a member of the group is copied to your directory.

"remark"

Specifies a remark string to be logged in the history file with this command, usually to explain why the command was entered. Specifying the **remark** parameter is equivalent to specifying the /REMARK qualifier after the FETCH keyword.

DESCRIPTION

The FETCH command copies the specified generation of a module from the VDE library to a file in your default directory or the directory specified with the /OUTPUT qualifier. The module is not reserved from the library nor can it be replaced into the VDE library. You can fetch a module whether or not it is reserved.

The presence or absence of a remark determines whether the FETCH transaction is recorded in the CMS history file for the facility. If you enter a remark, VDE records the transaction in the history file. If you enter a null remark, VDE does not record the transaction in the history file.

When you fetch a generation of a module from a VDE library, VDE restores the file's creation time, revision time, record format, and record attributes. The file that is placed in your directory has the same creation and revision times as the file that created the generation that you are fetching.

QUALIFIERS

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each module fetched. The /CONFIRM qualifier causes VDE to print a message for each module generation that you have specified asking whether you want to fetch that generation. If you answer YES (or Y), the module is fetched. If you answer NO (or N), the module is not fetched. The /NOCONFIRM qualifier causes VDE to fetch each specified module generation without asking for confirmation.

/GENERATION=gen-expr

Specifies that the module generation with the CMS generation expression given by the **gen-expr** parameter be fetched from the VDE library. When this qualifier is specified, the generation number specified on the **gen-spec** parameter is ignored.

/HISTORY["string"]

Specifies that history information is to be included in the file placed in your directory. The history of the module lists the generations that preceded the current generation of the module. Each generation produces one line of history information in the output file. This line gives the CMS generation number, user name, date, time, and remark associated with the generation.

The quoted string specifies the format of the history line. The quoted string must contain the characters #H or #B (lower case is allowed) and can contain other printing characters (such as comment delimiters). Each history line is formed by inserting the history information where you specify #H or #B in the history string. If you specify #H, the history information is inserted at the end of the file and if you specify #B it is inserted at the beginning of the file. To include a quotation mark in the output history string, type it twice (`""`). To include a number sign (#) in

the output history string, type it twice (##). If the **string** parameter is omitted, VDE uses the module's default history string.

If the /HISTORY and /NOHISTORY qualifiers are both omitted, the FETCH command generates history information only if the history attribute is enabled for the module. This attribute is set with the CREATE MODULE or MODIFY MODULE command.

/NOHISTORY

Specifies that history information is not to be appended to the output file placed in your directory.

/LOG (default)
/NOLOG

Controls whether log messages are printed after each module generation is fetched. The /LOG qualifier causes such messages to be printed and /NOLOG suppresses them. The messages indicate that the modules have been fetched and that the database transaction has successfully committed.

/MERGE=stream-name
/MERGE=(gen-expr)

Controls whether another generation of the module (called the merge generation) is to be merged with the generation that is being fetched (called the retrieved generation). If you specify the /MERGE qualifier with the **stream-name** parameter, the merge generation is the latest generation of the module in the specified stream. If you specify the /MERGE qualifier with the **gen-expr** parameter enclosed in parentheses, the merge generation is the generation with the specified generation expression (for example, ''2A3'').

When you specify the /MERGE qualifier, VDE merges the lines of the retrieved generation with the lines of the merge generation and delivers the merged text as a single file to your output directory. The file that is placed in your directory has the current creation and revision times. The merge generation cannot be on the same line of descent as the retrieved generation. When there is a conflict between blocks of one or more lines, VDE includes the conflicting lines and flags the conflict.

VDE uses the callable interface to CMS to provide the merge capability. For a detailed explanation of how two generations are merged and how CMS (and therefore VDE) treats conflicts between the generations, see the CMS documentation.

/NOTES["string"]

Specifies that CMS notes are to be appended to the lines of the output file. CMS notes indicate in what module generation each source line was created.

The quoted string specifies the format of the note. The quoted string can contain text (including the appropriate comment delimiter) and the characters #G or #g. The CMS generation expression of the generation in which the line first appeared in inserted where you specify #G or #g. The resulting string is appended to the end of the output line. To include a quotation mark in the output notes string, type it twice (""). To include a number sign (#) in the output notes string, type it twice (##). If the **string** parameter is omitted, VDE uses the module's default notes string.

(You set the default notes string with the CREATE MODULE or MODIFY MODULE command.)

If the /NOTES and /NONOTES qualifiers are both omitted, the FETCH command generates CMS notes only if the notes attribute is enabled for the module. This attribute is set with the CREATE MODULE or MODIFY MODULE command.

/NONOTES

Specifies that CMS notes are not to be appended to the lines of the file placed in your directory.

/NOOUTPUT

Specifies that the fetch operation is to be performed along with any history processing and error checking but that no output file is to be created.

/OUTPUT=file-specification

Directs VDE to write the fetched module to the specified file. If you omit the directory from the file specification, VDE writes the file to your default directory. If you omit the file name or file type, VDE uses the module name and type by default. If you omit the /OUTPUT and /NOOUTPUT qualifiers entirely, VDE creates a file in your current default directory with the same name as the module.

If you fetch more than one module with the same /OUTPUT qualifier (by using either wildcards or a group name in the module-name parameter) and you do not specify wildcards in the output file specification, VDE creates successive versions of the file indicated by the /OUTPUT qualifier.

/POSITION=column-number

Specifies the column in which CMS notes are to be placed. The column number must be an integer in the range 1 to 511. If the length of an output line is less than the specified column number, the note appears at the specified column number. If the length of the line is greater than or equal to the column number, the note is placed at the next tab stop after the end of the line. (Tab stops are at position 9 and every 8 characters thereafter.)

If the /POSITION qualifier is omitted, the default notes position for the module is used. The default notes position is set with the CREATE MODULE or MODIFY MODULE command. The /POSITION qualifier has no effect unless the /NOTES qualifier is specified or the notes attribute is set for the module.

/REMARK="string"

Specifies a remark string to be logged in the CMS history file by this command, usually to explain why the command was entered. The remark is enclosed in quotation marks. If no remark is entered, the fetch operation is not logged.

/STREAM=stream-name

Specifies that the modules be fetched from the development stream given by the **stream-name** parameter. If this qualifier is omitted, the modules are fetched from the default development stream. If this qualifier is omitted and no default stream is defined, VDE prompts you for the stream name.

EXAMPLES

1

```
VDE> FETCH [FACIL]MODNAM.MAR/REMARK="Just curious"
%VDE-I-FETCHED, module generation [FACIL]MODNAM.MAR;17(15A2) fetched from library
%VDE-I-OUTFILE, output file is DEV$:[MYDIR]MODNAM.MAR;
VDE>
```

This FETCH command retrieves the most recent generation of module [FACIL]MODNAM.MAR in the default development stream. The log message shows that the most recent generation is generation 17. Its CMS generation expression is 15A2. The file MODNAM.MAR is created in the user's default directory.

2

```
VDE> FETCH FOO.BAR;5
%VDE-I-FETCHED, module generation [CODE]FOO.BAR;5(5) fetched from library
%VDE-I-OUTFILE, output file is DEV$:[MYDIR]FOO.BAR;
VDE>
```

This example fetches generation 5 of module FOO.BAR in the default stream. Generation 5 is not necessarily the most recent generation of the module in the default stream. Since no facility name is specified, the module is assumed to be in the default facility, facility CODE.

3

```
VDE> FETCH FOO.BAR;-1
%VDE-I-FETCHED, module generation [CODE]FOO.BAR;5(5) fetched from library
%VDE-I-OUTFILE, output file is DEV$:[MYDIR]FOO.BAR;
VDE>
```

This example is the same as the previous example except that the user specifies a relative generation number, –1. This generation number causes VDE to fetch the generation before the most recent generation. In this case, generation 6 is the most recent generation. As a result, the FETCH command retrieves generation 5, as the log message shows.

4

```
VDE> FETCH MOD1, MOD2, MOD3, X*
%VDE-I-FETCHED, module generation [CODE]MOD1.C;4(4) fetched from library
%VDE-I-OUTFILE, output file is DEV$:[MYDIR]MOD1.C;
%VDE-I-FETCHED, module generation [CODE]MOD2.C;8(7C1) fetched from library
%VDE-I-OUTFILE, output file is DEV$:[MYDIR]MOD2.C;
%VDE-I-FETCHED, module generation [CODE]MOD3.C;1(1) fetched from library
%VDE-I-OUTFILE, output file is DEV$:[MYDIR]MOD3.C;
%VDE-I-FETCHED, module generation [CODE]MOD3.PLI;1(1) fetched from library
%VDE-I-OUTFILE, output file is DEV$:[MYDIR]MOD3.PLI;
%VDE-I-FETCHED, module generation [CODE]XAVIER.PAS;25(5A2B8) fetched from library
%VDE-I-OUTFILE, output file is DEV$:[MYDIR]XAVIER.PAS;
%VDE-I-FETCHED, module generation [CODE]XERXES.FOR;5(5) fetched from library
%VDE-I-OUTFILE, output file is DEV$:[MYDIR]XERXES.FOR;
VDE>
```

In this example, the FETCH command retrieves the latest generations of modules MOD1.C, MOD2.C, MOD3.C, and MOD3.PLI, along with all source modules whose names begin with the letter X. The module type defaults to the wildcard string that matches all module types (*). Hence the command fetches both MOD3.C and MOD3.PLI.

HELP

The HELP command allows users to obtain command syntax and functional information about the VDE and the VSC utilities.

FORMAT

HELP *[topic]*

PARAMETER

topic
Specifies the topic or command name about which you want information.

DESCRIPTION

To display the list of all available VDE commands and topics, enter the HELP command without a parameter. To obtain information about individual commands or topics, enter HELP followed by the command or topic name.

QUALIFIERS

None.

EXAMPLES

1
```
VDE> HELP
```

This command causes VDE to list all commands and topics for which you can request help text.

2
```
VDE> HELP BUILD
```

This command causes VDE to display help text explaining the BUILD command.

3
```
VDE> HELP CREATE MODULE/REMARK
```

This command causes VDE to display help text explaining the /REMARK qualifier to the CREATE MODULE command.

4
```
VDE> HELP LOGICAL_NAMES
```

This command causes VDE to display help text explaining logical names and how they are used by VDE.

INSERT GENERATION

Inserts a module generation into a specified development stream, making it the latest generation of the module in that stream.

Requires PERFREP privilege.

FORMAT

INSERT GENERATION *gen-spec [, gen-spec...] targ-stream-name ["remark"]*

PARAMETER

gen-spec

Specifies a module generation in a specified development stream that should be inserted into the stream given by the **stream-name** parameter. A generation specification consists of the following in order:

- an optional facility name enclosed in square brackets
- a module name
- an optional type name preceded by a period
- an optional generation number preceded by a semicolon

An example of a generation specification is [RMS]MOD1.MAR;5.

If you omit the facility name, the default facility is assumed. If you omit the type name, generations for all source modules with the specified module name in the specified facility are inserted into the given stream. If you omit the generation number, the most recent generation of the module in the stream specified with the /STREAM qualifier is inserted into the given stream.

If you specify a zero or negative generation number, that number is relative to the most recent generation of the module in the stream given by the /STREAM qualifier. For example, if the most recent generation is generation 5, then generation 0 is the same as generation 5 and generation –1 is the same as generation 4.

If you specify the /GENERATION qualifier, VDE inserts the generation with the given CMS generation expression and ignores the generation number on the **gen-spec** parameter.

You can insert more than one generation by using wildcard characters in the facility name, module name, or type name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Wildcard characters are not allowed in the generation number. Those module generations whose names match the wildcard pattern are inserted into the **stream-name** stream.

Unless the WILD_INS_GEN library attribute is enabled, VDE does not allow wildcard characters in the generation specification when you use the /GENERATION qualifier. It seldom makes sense to select generations with the same CMS generation expression from different modules. For the same reason, VDE does not allow you to use wildcard characters and

a non-zero generation number in the same generation specification (as in *.*;–2), again unless the WILD_INS_GEN library attribute is set.

targ-stream-name

Specifies the development stream into which the generations given by the **gen-spec** parameters should be inserted.

One can specify the source generation for the operation in several ways, one of which is based on the generation in a specified source stream—see the /STREAM qualifier.

"remark"

Specifies a remark string that explains why you inserted the generation into the specified stream. The quoted remark string can be up to 132 characters long. Specifying the **remark** parameter is equivalent to specifying the /REMARK qualifier after the INSERT GENERATION keywords.

DESCRIPTION

The INSERT GENERATION command inserts the module generations you specify with the **gen-spec** parameters into the development stream you specify with the **stream-name** parameter. Each such generation becomes the latest generation of the corresponding module in the **stream-name** stream. The command thus lets you move module generations from one stream to another stream.

If the specified module is currently reserved in the specified stream, VDE terminates that reservation since it is based on a module generation that no longer is the latest generation for the stream. VDE sends a mail message to notify the owner of the reservation that the reservation is lost.

Note that there are two stream specifications possible on this command. The destination stream—specified by the target stream name parameter—and the optional source stream qualifier.

QUALIFIERS

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each generation inserted into the specified stream. The /CONFIRM qualifier causes VDE to print a message for each generation giving the name of the generation and asking whether you want to insert the generation into the stream. If you answer YES (or Y), the generation is inserted into the stream. If you answer NO (or N), the generation is not inserted into the stream. The /NOCONFIRM qualifier causes VDE to insert each generation into the specified stream without asking for confirmation.

/NOCMS_CLASS

Prevents the inserted generations from being inserted into the CMS class for the specified stream, even if a CMS class is normally maintained for the stream. The INSERT GENERATION command executes faster than otherwise if CMS class memberships are not updated.

/GENERATION=gen-expr

Specifies that the generation with the CMS generation expression given by the **gen-expr** parameter be inserted into the specified stream. When this qualifier is specified, the generation number specified on the **gen-spec** parameter is ignored.

/LOG (default)
/NOLOG

Controls whether log messages are printed after the generations are inserted into the stream. The /LOG qualifier causes such messages to be printed and /NOLOG suppresses them. The messages indicate that the generations have been inserted into the specified stream and that the database transaction has successfully committed.

/REMARK="string"

Specifies a remark string that explains why you inserted the generation into the specified stream. The quoted remark string can be up to 132 characters long.

/STREAM=stream-name

Specifies the stream in which the generation should be looked up if you omit the /GENERATION qualifier. This qualifier thus affects the interpretation of the generation number in the command-level **gen-spec** parameter when the /GENERATION qualifier is omitted. If you omit both the /STREAM qualifier and the /GENERATION qualifier, VDE prompts you for the name of the stream in which to look up the generation number.

EXAMPLES

1

```
VDE> INSERT GENERATION [FACIL]MODNAM.C;4/STREAM=V2.0 BL-22
%VDE-I-GENINSERT, generation [FACIL]MODNAM.C;4(4) inserted into stream BL-22
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
VDE> SET STREAM BL-22
VDE> SHOW GENERATION [FACIL]MODNAM.C
Generation [FACIL]MODNAM.C;4(4)       by user JONES        on  9-MAY-1990 19:43
VDE>
```

This example inserts generation 4 of module [FACIL]MODNAM.C from stream V2.0 into stream BL-22, making that generation the latest generation in stream BL-22.

INVOKE SCRIPT

Invokes a VDE script from the VDE database and writes the resulting DCL command file to a directory in the current VDE library or to a VDE scratch directory.

FORMAT INVOKE SCRIPT

DESCRIPTION

The INVOKE SCRIPT command reads a specified VDE script from the VDE database, executes the script language in the script to produce a DCL command file, and writes that DCL command file to the appropriate directory. The INVOKE SCRIPT command does not execute the resulting DCL command file.

The directory that receives the DCL command file is the directory that would receive it if the script were executed normally as part of a VDE command. This directory depends on the script type, and is displayed in a log message. You can use the INVOKE SCRIPT command to examine the DCL code generated by VDE in response to your scripts.

You must specify a script-type qualifier to select the script you want invoked. See the description of the CREATE SCRIPT command for an explanation of the format of scripts and the different types of scripts. Also see the description of the CREATE SCRIPT command for an explanation of how VDE selects scripts for specific modules or facilities. If the scripts you select have multiple segment numbers, VDE selects one script for each segment number and generates the DCL command file from those scripts in order by their segment numbers.

The INVOKE SCRIPT command always selects and invokes the desired script in the context of the default development stream.

Keywords used within scripts are documented in Script Keywords, while script functions are in Script Functions, and script types are in Script Types. For other script-related commands, also see CREATE SCRIPT, DELETE SCRIPT, EXTRACT SCRIPT, MODIFY SCRIPT, and SHOW SCRIPT.

QUALIFIERS

/BUILD

Specifies that the default stream's BUILD script be invoked.

/COMPILE=mod-name

Specifies that a COMPILE script for the default stream be invoked for a specified module, using VDE's normal script selection rules. The **mod-name** parameter specifies the name of the module. This name may not contain wildcard characters. The specified module determines the values of the $FAC, $MOD, and $TYP script functions (among others) when the script is invoked.

/COPY=mod-name

Specifies that a COPY script for the default stream be invoked for a specified module, using VDE's normal script selection rules. The **mod-name** parameter specifies the name of the module. This name may not contain wildcard characters. The specified module determines the values of the $FAC, $MOD, and $TYP script functions (among others) when the script is invoked.

/CREATE_FACILITY

Specifies that the facility creation script be invoked. The the values of the $FAC, $MOD, and $TYP script functions (among others) are determined when the script is invoked.

/DELETE_GENERATION=mod-name

Specifies that a DELETE_GENERATION script for the default stream be invoked for a specified module, using VDE's normal script selection rules. The **mod-name** parameter specifies the name of the module. This name may not contain wildcard characters. The specified module determines the values of the $FAC, $MOD, and $TYP script functions (among others) when the script is invoked.

/DIFFERENCES=mod-name

Specifies that a DIFFERENCES script for the default stream be invoked for a specified module, using VDE's normal script selection rules. The **mod-name** parameter specifies the name of the module. This name may not contain wildcard characters. The specified module determines the values of the $FAC, $MOD, and $TYP script functions (among others) when the script is invoked.

/FETCH=mod-name

Specifies that a FETCH script for the default stream be invoked for a specified module, using VDE's normal script selection rules. The **mod-name** parameter specifies the name of the module. This name may not contain wildcard characters. The specified module determines the values of the $FAC, $MOD, and $TYP script functions (among others) when the script is invoked.

/INSERT_GENERATION=mod-name

Specifies that an INSERT_GENERATION script for the default stream be invoked for a specified module, using VDE's normal script selection rules. The **mod-name** parameter specifies the name of the module. This name may not contain wildcard characters. The specified module determines the values of the $FAC, $MOD, and $TYP script functions (among others) when the script is invoked.

/LINK=mod-name

Specifies that a LINK script for the default stream be invoked for a specified module, using VDE's normal script selection rules. The **mod-name** parameter specifies the name of the module. This name may not contain wildcard characters. The specified module determines the values of the $FAC, $MOD, and $TYP script functions (among others) when the script is invoked.

/LOG (default)
/NOLOG

Controls whether log messages are printed after the script is invoked. The /LOG qualifier causes such messages to be printed and the /NOLOG qualifier suppresses them. These messages indicate that the script has been invoked and give the name of the resulting DCL command file.

/NEW_STREAM=fac-name

Specifies that a NEW_STREAM script for the default stream be invoked for a specified facility, using VDE's normal script selection rules. The **fac-name** parameter specifies the name of the facility. This name may not contain wildcard characters. The specified facility determines the value of the $FAC script function when the script is invoked.

/NOKEEP=mod-name

Specifies that a NOKEEP script for the default stream be invoked for a specified module, using VDE's normal script selection rules. The **mod-name** parameter specifies the name of the module. This name may not contain wildcard characters. The specified module determines the values of the $FAC, $MOD, and $TYP script functions (among others) when the script is invoked.

/REPLACE=mod-name

Specifies that a REPLACE script for the default stream be invoked for a specified module, using VDE's normal script selection rules. The **mod-name** parameter specifies the name of the module. This name may not contain wildcard characters. The specified module determines the values of the $FAC, $MOD, and $TYP script functions (among others) when the script is invoked.

/RESERVE=mod-name

Specifies that a RESERVE script for the default stream be invoked for a specified module, using VDE's normal script selection rules. The **mod-name** parameter specifies the name of the module. This name may not contain wildcard characters. The specified module determines the values of the $FAC, $MOD, and $TYP script functions (among others) when the script is invoked.

/SET_FACILITY=fac-name

Specifies that a SET_FACILITY script for the default stream be invoked for a specified facility, using VDE's normal script selection rules. The **fac-name** parameter specifies the name of the facility. This name may not contain wildcard characters. The specified facility determines the value of the $FAC script function when the script is invoked.

/SET_STREAM

Specifies that the SET_STREAM script for the default stream be invoked.

/STAGING=mod-name

Specifies that a STAGING script for the default stream be invoked for a specified module, using VDE's normal script selection rules. The **mod-name** parameter specifies the name of the module. This name may not contain wildcard characters. The specified module determines the values of the $FAC, $MOD, and $TYP script functions (among others) when the script is invoked.

/UNRESERVE=mod-name

Specifies that an UNRESERVE script for the default stream be invoked for a specified module, using VDE's normal script selection rules. The **mod-name** parameter specifies the name of the module. This name may not contain wildcard characters. The specified module determines the values of the $FAC, $MOD, and $TYP script functions (among others) when the script is invoked.

EXAMPLES

1

```
$ INVOKE SCRIPT/COMPILE=MODNAM.C
%VDE-I-SCRPUSED, script to compile module [CODE]*.C (segment 10) used
%VDE-I-SCRPFILE, script produced DCL command file DEV$:[PROJDIR.MAIN.CODE
.VDE$COM]MODNAM.C;12
 script to compile module [CODE]*.C
VDE> EXIT
$
$ TYPE [PROJDIR.MAIN.CODE.VDE$COM]MODNAM.C
$ ! DCL command file to execute a VDE script
$ !
$ ON WARNING THEN GOTO VDE$ERROR
$ !
$ SET VERIFY
$ DEFINE/NOLOG SRC$ DEV$:[PROJDIR.MAIN.CODE.SRC]
$ DEFINE/NOLOG OBJ$ DEV$:[PROJDIR.MAIN.CODE.OBJ]
$ CC/DEBUG/OBJ=OBJ$:/ANA=OBJ$: SRC$:MODNAM.C
$ VDE/DEPEND C OBJ$:MODNAM.ANA
$ !
$ SET NOVERIFY   ! Successful script completion
$ VDE/STATUS SUCCESS
$ EXIT
$VDE$ERROR:
$ SET NOVERIFY   ! Script completed with errors
$ VDE/STATUS FAILURE
$ EXIT
$
```

In this example, the user invokes the script that compiles module MODNAM.C in facility CODE. VDE selects the appropriate script using its normal script selection rules and applies the script to that module. VDE writes the resulting DCL command file to a file called MODNAM.C in the VDE$COM subdirectory for the given stream and facility. (The COMPILE and BUILD commands use this directory when they generate and execute build-step command files.) The subsequent DCL TYPE command shows the text of the DCL command file. This text includes some standard ''boiler-plate'' at the beginning and end of the file. In the middle of the file are the DCL commands that come from the script itself. These commands enable command-line verification, define two logical names, compile the module MODNAM.C, and use the VDE/DEPEND command to record the compile step's build dependencies. The surrounding boiler-plate traps errors and records the build step's status (success or failure).

LINK

Links the specified modules using the appropriate scripts from the database.

Requires BUILD privilege.

FORMAT

LINK *mod-name [, mod-name...]*

PARAMETER

mod-name

Specifies a module to be linked. The module name consists of an optional facility name enclosed in square brackets, a module name, and an optional type name preceded by a period, such as [FACIL]MOD1.EXE. If no facility name is specified, the default facility is assumed. If no type name is specified, module type EXE is assumed.

You can link more than one module by using wildcard characters in any of the three components of the module name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. VDE links modules whose names match the wildcard pattern.

DESCRIPTION

The LINK command links a module the same way it is linked in a build job. The LINK command retrieves the link script for each specified module and generates a DCL command file from the script. The command then executes the command file in a subprocess. If the module links successfully, VDE updates the build dependency information for the link step in the database.

The LINK command can be used to link a module without starting a build job. It is often used to test link scripts. It can also be used to collect the initial dependency information for a link step that does not yet have dependency information in the database.

QUALIFIERS

/LOG (default)
/NOLOG

Controls whether log messages are printed after the link step executes. The /LOG qualifier causes such messages to be printed and the /NOLOG qualifier suppresses them. These messages indicate that the link step has completed, either successfully or with errors, and that the dependency information for the step has been updated in the VDE database.

/STREAM=stream-name

Specifies that the modules be linked for the stream given by the **stream-name** parameter. If this qualifier is omitted, the modules are linked for the default stream.

EXAMPLES

1

```
VDE> LINK [CODE]X.EXE
$ DEFINE/NOLOG OBJ$ DEV$:[PROJDIR.MAIN.CODE.OBJ]
$ LINK /EXE=OBJ$:X.EXE /MAP=OBJ$:X.MAP -
        OBJ$:OBJLIB.OLB/INCLUDE=C,OBJ$:OBJLIB.OLB/LIBRARY
$ !
$ SET NOVERIFY   ! Successful script completion
%VDE-I-LINKSUCC, link of [CODE]X.EXE completed successfully
%VDE-I-DEPUPDATE, dependency information updated for build step
VDE>
```

This example links module X.EXE in facility CODE. The output from the execution of the link script is displayed on the terminal. The final log messages show that the link step completed successfully and that its dependency information was updated in the database.

MODIFY ARCHITECTURE

Modifies one or more architecture table entries in the VDE database. Each architecture table entry...

Requires the CREARCH privilege.

ARCHITECTURE support is not fully implemented.

An architecture is a construct that encompasses one or more facilities, and allows modules to exist in a "common" or "default" facility, and zero or more architecture specific facilities. In must the same fashion as the SYS$SYSROOT: logical name allows one to support both system-specific and cluster-common files, an architecture allows one to support architecture-specific and architecture-common modules.

FORMAT

MODIFY ARCHITECTURE *arch-name [, arch-name...]*

PARAMETER

arch-name

The name of the architecture to be modified. This name may be up to 39 characters long and must follow VDE name syntax.

DESCRIPTION

Under Construction...

QUALIFIERS

/CONFIRM (default)
/NOCONFIRM

Under Construction...

/DELETE (default)
/NODELETE

Under Construction...

/LOG (default)
/NOLOG

Under Construction...

/NAME

Under Construction...

/REMARK="string"

Specifies a remark string to be stored for the modified architecture. The quoted remark string, which can be up to 132 characters long, may contain any explanatory remark about the architecture. that you want stored in the VDE database. Typically the remark describes the function or purpose of the architecture.

MODIFY ARCHITECTURE

EXAMPLES

1
```
VDE> MODIFY ARCHITECTURE VAXPDP/REMARK="Version n"
VDE>
```

Under Construction...

MODIFY BUILD_JOB

Modifies the batch queues, process count, remark string, and other attributes of existing build jobs in the VDE database.

Requires BUILD privilege.

FORMAT

MODIFY BUILD_JOB *[stream-name [, stream-name...]]*

PARAMETER

stream-name

The name of a development stream containing the build jobs to be modified. To specify the build jobs, use the /IDENTIFICATION qualifier. If you do not use the /IDENTIFICATION qualifier, the most recent build job for the stream is modified.

You can modify build jobs in more than one stream by using wildcard characters in the stream name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Build jobs in those streams whose names match the wildcard pattern are modified.

If you omit the **stream-name** parameter, VDE modifies the attributes of one or more build jobs for the default development stream.

DESCRIPTION

The MODIFY BUILD_JOB command modifies the attributes of build jobs having the specified stream names and build job identification numbers. You can modify the following attributes: the batch queues to which the job is submitted for execution, the total number of batch processes submitted, the remark string, the step-delay interval, and the timeout interval. VDE modifies these attributes by updating the database with the information you specify. You cannot modify the queues or process count of a build job that has started running or that has completed execution. You can, however, suspend a running build job, modify those attributes, and restart the job.

QUALIFIERS

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each build job modified. The /CONFIRM qualifier causes VDE to print a message for each build job asking whether you want that build job modified. If you answer YES (or Y), the attributes of the build job are modified in the database. If you answer NO (or N), the attributes are not modified. The /NOCONFIRM qualifier causes VDE to modify the attributes of the specified build jobs without asking for confirmation.

/IDENTIFICATION=id-range

Specifies the build job identification numbers of the build jobs to be modified for the specified development stream. The **id-range** parameter can be a single integer value, a range of integers, or a wildcard character. If the parameter is a single integer value (for example, /IDENT=5), VDE modifies the build job with that identification number. If the parameter is a range of integers (specified as two build job identification numbers separated by a colon as in /IDENT=3:5), VDE modifies all build jobs with identification numbers in that range. Finally, if the **id-range** parameter is the asterisk wildcard (as in /IDENT=*), VDE modifies all build jobs for the specified stream.

If you specify a zero or negative build job identification number in the **id-range** parameter, that number is relative to the identification number of the most recent build job for the stream. For example, if the most recent build job is job 5, then /IDENT=0 is equivalent to /IDENT=5 and /IDENT=–1 is equivalent to /IDENT=4.

If you do not use the /IDENTIFICATION qualifier, VDE modifies the attributes of the most recent build job for the specified stream.

/LOG (default)
/NOLOG

Controls whether log messages are printed after each build job is modified. The /LOG qualifier causes such messages to be printed and the /NOLOG qualifier suppresses them. These messages indicate that the build job has been modified in the database and that the database transaction has successfully committed.

/PROCESS_COUNT=proc-count

Specifies the number of processes to execute the build job. When the build job is started with a START BUILD_JOB command, VDE creates the processes by submitting batch jobs to the queues specified by the /QUEUE qualifier or to the default queues for the default stream. The number of processes that VDE creates (that is, the number of OpenVMS batch jobs it submits) is specified by the **proc-count** parameter. As a result, this parameter determines the degree of parallelism for the build job.

/QUEUE=(queue-name [,queue-name...])

Specifies the OpenVMS batch queues to which the build job is submitted by a subsequent START BUILD_JOB command. Each **queue-name** parameter specifies the name of one batch queue. The **queue-name** parameter may optionally be followed by a process count in parentheses. This count specifies the number of processes that are started on the queue to run the build job. For example, SYS$BATCH(3) causes three build-job processes to be submitted to queue SYS$BATCH. If the process count is omitted, one process is submitted to the queue.

If you only specify one **queue-name** parameter, the parentheses may be omitted.

If you also use the /PROCESS_COUNT qualifier, that qualifier states the total number of build-job processes to create. The process counts attached to the **queue-name** parameters on the /QUEUE qualifier are then used as scaling factors to distribute build-job processes among the queues proportionally. For example, if you specify queues HI_BATCH, HO_BATCH(3), and a total process count of eight, then two processes are submitted to queue HI_BATCH and six processes to queue HO_BATCH. If

the total process count does not divide evenly into the sum of the scaling factors, the remaining processes are allocated to the queues in the order the queues are listed. If the total process count is nine, for example, the one extra process is allocated to queue HI_BATCH because HI_BATCH is listed first.

/REMARK="string"

Specifies a remark string to be stored in the database for each specified build job. The quoted remark string, which can be up to 132 characters long, may contain any explanatory remark that you want to store in the database. This string replaces the remark string from the original BUILD command or from any previous MODIFY BUILD_JOB command.

/STEP_DELAY=time-interval

Specifies the step-delay time interval used by VDE$BUILD. VDE$BUILD is the program that executes the build steps belonging to the build job. If VDE$BUILD cannot find an available build step in the database, the program waits the amount of time specified by the /STEP_DELAY qualifier before it looks in the database again. See the description of the BUILD command for a more detailed description of this qualifier.

/TIMEOUT=time-interval

Specifies the timeout interval for the build job. If no build steps complete during this interval, the VDE$BUILD utility stops the build job and assigns it TIMEOUT status. Therefore, the timeout interval must be longer than the time required to complete the longest build step. See the description of the BUILD command for a more detailed description of this qualifier.

EXAMPLES

1
```
VDE> MODIFY BUILD_JOB V2.0-3/IDENT=15/REMARK="New remark string"
%VDE-I-BLDJOBMOD, build job 15 for stream V2.0-3 modified in database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example modifies the remark string of build job 15 for stream V2.0-3.

2
```
VDE> MODIFY BUILD_JOB/QUEUE=(PHI_BATCH, FOO_BATCH(3))/PROCESS_COUNT=10
%VDE-I-BLDJOBMOD, build job 12 for stream V5.2 modified in database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This command modifies the batch queues and process count associated with the most recent build job for the default stream, build job 12 of stream V5.2.

MODIFY CONTEXT

Modifies the attributes of one or more contexts in the VDE database. The attributes of a context that you can modify include its name, remark, default development stream, and default facility. One context can be used to set the default stream and facility when first connecting to a VDE database.

Requires the CRECTX privilege to modify public contexts.

FORMAT

MODIFY CONTEXT *ctx-name [, ctx-name...]*

PARAMETER

ctx-name

The name of an existing context whose attributes are to be modified. You can modify more than one context at a time by using wildcard characters in the context name. The percent sign (%) in a context name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Those contexts whose names match the wildcard pattern are modified.

DESCRIPTION

A ‘‘context’’ specifies defaults for VDE sessions. When you modify a context using the MODFY CONTEXT command, you specify the name of the context, the associated development stream, and the associated facility.

To apply the defaults associated with a context, specify the context name as a parameter to the SET CONTEXT command. To set a particular context as the default context for use when you issue the SET LIBRARY command, use the SET CONTEXT/DEFAULT command to mark a particular context as the default context.

There are two kinds of contexts available:

- the *private context* is available to and only visible to the user who created it.
- the *public context* is available to all users of the current VDE library.

You do not need special VDE privileges to create or modify a private context, but you must have the VDE CRECTX privilege to create or modify a public context. While a private context can have the same name as a public context, neither two private contexts nor two public contexts can share the same name.

QUALIFIERS

/ARCHITECTURE

Under Construction...

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each context definition modified. The /CONFIRM qualifier causes VDE to print a

message for the specified context giving the name of the context and asking whether you want to modify that context. If you answer YES (or Y), the attributes are modified. If you answer NO (or N), the attributes are not modified. The /NOCONFIRM qualifier causes VDE to modify the attributes of the context without asking for confirmation.

/FACILITY=facil-name
/NOFACILITY

The /FACILITY qualifier specifies that the indicated facility becomes the default facility when the context is established with the SET CONTEXT command. The default facility is the facility that is assumed when the facility name is omitted from a module name. The /NOFACILITY qualifier specifies that the default facility established by the context is null.

/LOG (default)
/NOLOG

Controls whether log messages are printed after the context definitions are modified. The /LOG qualifier causes such messages to be printed and /NOLOG suppresses them. The messages indicate that the context definitions have been modified and that the database transaction has successfully committed.

/NAME=ctx-name

Specifies a new name for the context. The new context name can be up to 39 characters long and must follow VDE name syntax. After the context name is changed, you must refer to the context by its new name in all subsequent VDE commands. The old name is lost from the database.

/PRIVATE (default)
/PUBLIC

Specifies whether the context that is modified is private or public. The /PRIVATE qualifier causes VDE to modify your private context by that name. The /PUBLIC qualifier causes VDE to modify the public context by that name. You must have the CRECTX privilege to use the /PUBLIC qualifier.

Of /PRIVATE and /PUBLIC, /PRIVATE is the default.

/REMARK="string"

Specifies a remark string for the context. This string replaces the remark string from the original CREATE CONTEXT command or from any previous MODIFY CONTEXT command. The quoted remark string, which can be up to 132 characters long, may contain any explanatory remark about the context that you want stored in the VDE database. Typically the remark describes the function or purpose of the context.

/STREAM=stream-name
/NOSTREAM

The /STREAM qualifier specifies that the indicated development stream becomes the default stream when the context is established with the SET CONTEXT command. The /NOSTREAM qualifier specifies that the default stream established by the context is null.

EXAMPLES

1

```
VDE> MODIFY CONTEXT DEFAULT/STREAM=V2.2-BL1/FACILITY=COPY
%VDE-I-CTXMOD, information for context DEFAULT modified in the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE> DEFINE/KEY KP6 "SET CONTEXT DEFAULTS" /TERMINATE
VDE> KP6
VDE> SHOW DEFAULTS
Library LIBNAME in root directory DEV$:[PROJDIR]
Default directory is DEV$:[SMITH]
Context DEFAULT (private to user SMITH)
   Default stream is V2.2-BL1
   Default facility is COPY

VDE>
```

The MODIFY CONTEXT command specifies stream V2.2-BL1 as the default stream and facility COPY as the default facility for private context DEFAULT. This context is the default at the start of each VDE session. The SET CONTEXT command establishes these defaults for the current VDE session, as the SHOW DEFAULTS command confirms.

2

```
VDE> MODIFY CONTEXT/PUBLIC/FACILITY=EXEC PUBCTX
%VDE-I-CTXMOD, information for context PUBCTX modified in the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE> SHOW CONTEXT PUBCTX
Context PUBCTX (public to all users)
   Default stream is V2.0
   Default facility is EXEC

VDE>
```

The MODIFY CONTEXT command establishes facility EXEC as the default facility for public context PUBCTX. The SHOW CONTEXT command shows that the change has taken effect.

MODIFY FACILITY

Modifies the attributes of one or more facilities in the VDE database. The attributes of a facility you can modify are its name and remark string.

Requires CREFAC privilege.

FORMAT

MODIFY FACILITY *facil-name [, facil-name...]*

PARAMETER

facil-name

The name of a facility in the current VDE library whose attributes are to be modified. You can modify more than one facility at a time by using wildcard characters in the facility name. The percent sign (%) in a facility name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. VDE modifies the attributes of those facilities whose names match the wildcard pattern.

QUALIFIERS

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each facility modified. The /CONFIRM qualifier causes VDE to print a message for the specified facility giving the name of the facility and ask whether you want to modify that facility. If you answer YES (or Y), the attributes are modified. If you answer NO (or N), the attributes are not modified. The /NOCONFIRM qualifier causes VDE to modify the attributes of the facility without asking for confirmation.

/DELETE
/NODELETE (default)

Sets or clears the **delete** attribute for the facility. This attribute allows the facility to be deleted with the DELETE FACILITY command. The /DELETE qualifier sets the delete attribute so that the facility can be deleted. The /NODELETE qualifier clears the delete attribute so that the facility cannot be deleted without first resetting the attribute.

/LOG (default)
/NOLOG

Controls whether log messages are printed after the facility definitions are modified. The /LOG qualifier causes such messages to be printed and /NOLOG suppresses them. The messages indicate that the facility definitions have been modified and that the database transaction has successfully committed.

/NAME=facil-name

Specifies a new name for the facility. The new facility name can be up to 39 characters long and must follow normal VDE name syntax. After the facility name has been changed, you must refer to the facility by its new

name in all subsequent VDE commands. The old name is lost from the database.

/PROPAGATE
/NOPROPAGATE

Specifies whether automatic change propagation is allowed by default for new modules created in this facility. /PROPAGATE causes new modules added to the facility to be marked as allowing automatic change propagation when reserved and replaced. /NOPROPAGATE marks such modules as not allowing change propagation. When you create new modules, you can override these defaults with qualifiers to the CREATE MODULE command.

/REMARK="string"

Specifies a remark string for the facility. This string replaces the remark string from the original CREATE FACILITY command or from any previous MODIFY FACILITY command. The quoted remark string, which can be up to 132 characters long, may contain any explanatory remark about the facility that you want stored in the VDE database. Typically the remark describes the function or contents of the facility.

EXAMPLES

1
```
VDE> MODIFY FACILITY ACCTNG /REMARK="New remark string"
%VDE-I-FACMOD, information for facility ACCTNG modified in the database
%VDE-I-COMMIT, database transaction has successfully committed
```

In this example, the user specifies a new remark string for facility ACCTNG. This remark string can be retrieved with a subsequent SHOW FACILITY command.

2
```
VDE> MODIFY FACILITY/NAME=ACCOUNTING ACCTNG
%VDE-I-FACMOD, information for facility ACCOUNTING modified in the database
%VDE-I-COMMIT, database transaction has successfully committed
```

In this example, the name of facility ACCTNG is changed to ACCOUNTING. You must refer to the facility by this new name in all subsequent VDE commands.

3
```
VDE> MODIFY FACILITY/CONFIRM ACCTNG/REMARK="New remark string", -
_VDE>                             FUMBLE/REMARK="New remark string"
Modify facility ACCTNG ? [No]: Y
%VDE-I-FACMOD, facility ACCTNG modified in the database
Modify facility FUMBLE ? [No]: N
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example illustrates the /CONFIRM qualifier. This qualifier causes VDE to ask the user whether each specified facility should be modified. For facility ACCTNG, the user answers Y (for YES) and the remark string for that facility is modified in the database, as the log message shows. For facility FUMBLE, the user answers N (for NO) and that facility is not modified. Hence there is no log message for that facility. The final log message indicates that the database transaction for the whole command has successfully committed.

MODIFY FOLD

Modifies the remark string of one or more fold records in the VDE database.

Requires RESREP privilege.

FORMAT

MODIFY FOLD *mod-name [, mod-name...]*

PARAMETER

mod-name

Specifies a module for which a fold record is to be modified. The module name consists of an optional facility name enclosed in square brackets, a module name, and an optional type name preceded by a period (such as [FACIL]MOD1.MAR). If the facility name is omitted, the module is assumed to belong to the default facility. If the type name is omitted, VDE modifies fold records for all source modules with the specified module name in the given facility.

You can modify fold records for more than one module by using wildcard characters in any of the three components of the module name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. VDE modifies fold records for all source modules whose names match the wildcard pattern.

DESCRIPTION

The MODIFY FOLD command modifies the remark string of each specified fold record in the VDE database.

QUALIFIERS

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each fold record modified. The /CONFIRM qualifier causes VDE to print a message for each fold record asking whether you want that record modified. If you answer YES (or Y), that fold record is modified in the database. If you answer NO (or N), the fold record is not modified. If you enter a module name with a wildcard specification, VDE asks you to confirm the modification of fold records for each module whose name matches the wildcard specification. The /NOCONFIRM qualifier causes VDE to modify the specified fold records without asking for confirmation.

/IDENTIFICATION=fold-ident

Specifies the fold record to be modified. This qualifier is required when you have multiple fold records for the same module in the same stream. The **fold-ident** parameter is the fold identifier of the fold record to be modified. The fold identifier is the identifier you specified or VDE assigned when you created the fold record, usually a small integer value. Use the SHOW FOLD command to determine the fold identifier of each fold record.

/LOG (default)
/NOLOG

Controls whether log messages are printed after each fold record has been modified. The /LOG qualifier causes such messages to be printed and the /NOLOG qualifier suppresses them. These messages indicate that the fold record has been modified and that the database transaction has successfully committed.

/REMARK="string"

Specifies a new remark string for the fold record. This string replaces the remark string from the original REPLACE or CREATE FOLD command or from any previous MODIFY FOLD command. The quoted remark string, which can be up to 132 characters long, may contain any explanatory remark about the fold record that you want stored in the VDE database. Typically the remark describes the purpose or contents of a source change to be folded into a development stream.

/STREAM=stream-name

Specifies the development stream for which fold records should be modified. The **stream-name** parameter gives the name of the stream. If this qualifier is omitted, VDE modifies fold records for the default development stream only.

/USERNAME=username

Specifies that fold records are to be modified on behalf of another user. The **username** parameter specifies the OpenVMS username of the other user. You must have the USERNAME privilege to use this qualifier.

EXAMPLES

1

```
VDE> MODIFY FOLD [CODE]X.FOR/IDENT=1/REMARK="New remark string"
%VDE-I-FOLDMOD, fold record 1 for module [CODE]X.FOR modified in database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example specifies a new remark string for fold record 1 for module [CODE]X.FOR. The log messages show that the fold record was successfully modified.

MODIFY GENERATION

Modifies the attributes associated with specific module generations in the VDE database.

Requires RESREP privilege.

FORMAT

MODIFY GENERATION *gen-spec [, gen-spec...]*

PARAMETER

gen-spec

Specifies a module generation in a specified (or the default) development stream whose attributes are to be modified. A generation specification consists of the following in order:

- an optional facility name enclosed in square brackets
- a module name
- an optional type name preceded by a period
- an optional generation number preceded by a semicolon

An example of a generation specification is [RMS]MOD1.MAR;5.

If you omit the facility name, module generations from the default facility are modified. If you omit the type name, generations for all modules with the specified module name in the specified facility are modified. If you omit the generation number, the most recent generation of the module in the specified stream is modified.

If you specify a zero or negative generation number, that number is relative to the most recent generation of the module in the stream specified with the /STREAM qualifier (or the default stream if that qualifier is omitted). For example, if the most recent generation is generation 5, then generation 0 is the same as generation 5 and generation –1 is the same as generation 4.

If you specify the /GENERATION qualifier, VDE modifies the generation with the given CMS generation expression and ignores the generation number on the **gen-spec** parameter.

You can modify more than one generation by using wildcard characters in the facility name, module name, or type name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. An asterisk (*) in place of the generation number matches all generations of a module. Those module generations whose names match the wildcard pattern are modified.

MODIFY GENERATION

DESCRIPTION

The MODIFY GENERATION command allows you to modify the attributes associated with a specific module generation in the VDE library.

Attributes that can be altered via the MODIFY GENERATION command include:

- The generation's remark string. This string is originally created by the CREATE MODULE or REPLACE command.
- The module's status in the specified development stream. This command can be used to mark a specific generation of the module as as obsolete in the specified development stream. An obsolete module remains in the VDE database since it may not be obsolete for old streams, only for new ones. Once a module is marked as obsolete for a given stream, that module is not carried over to any future stream that has the given stream as its parent.
- The module's status as the main line of descent in the the specified development stream.

QUALIFIERS

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want the attributes of each generation modified. The /CONFIRM qualifier causes VDE to print a message for each generation giving the name of the generation and asking whether you want to modify the generation. If you answer YES (or Y), the generation's attributes are modified. If you answer NO (or N), the attributes are not modified. The /NOCONFIRM qualifier causes VDE to modify the attributes of each generation without asking for confirmation.

/GENERATION=gen-expr

Specifies that the generation with the CMS generation expression given by the **gen-expr** parameter be modified. When this qualifier is specified, the generation number specified on the **gen-spec** parameter is ignored.

/LOG (default)
/NOLOG

Controls whether log messages are printed after the attributes of generations are modified. The /LOG qualifier causes such messages to be printed and /NOLOG suppresses them. The messages indicate that the generations have been modified and that the database transaction has successfully committed.

/MAIN_STREAM=stream-name

Specifies the stream that ''owns'' the main development thread for the specified generation. If the generation is later reserved and its successor is replaced into the stream given by the **stream-name** parameter (directly or through change propagation), the successor generation will get a ''whole-number'' CMS generation expression. If the current generation has generation expression 10, for example, the generation created by replacing into the specified stream gets generation expression 11. A replacement into any other stream gets a generation expression such as 10A1 or 10B1. Similarly, if the current generation has generation expression 5A2, the

specified stream gets the ‘‘whole-number’’ successor 5A3 instead of a generation expression such as 5A2A1.

VDE automatically maintains information about what stream owns the whole-number successor for each generation in the library. This information is determined by your use of the /MAIN qualifier to the CREATE STREAM command. For this reason, you will seldom need the /MAIN_STREAM qualifier to the MODIFY GENERATION command. However, there are exceptional cases where you may need to change what stream owns the whole-number successor for a given generation, and then this qualifier lets you set that attribute.

/NOTE=note-number

The **note-number** parameter specifies the note number for the (previously-entered) SCT note associated with this replacement.

VDE can also enter the note into the conference directly, see the /SCT qualifier for information.

/OBSOLETE
/NOOBSOLETE

Specifies whether the module is marked as obsolete in the specified development stream. The /OBSOLETE qualifier specifies that the module is obsolete in that stream. An obsolete module is not propagated to future streams created from that stream. The /NOOBSOLETE qualifier specifies that the module is not obsolete. If neither qualifier is specified, the current status of the module is not changed.

/REMARK="string"

Specifies a new remark string to be stored for the generation. This string replaces the generation’s current remark string. The quoted remark string, which can be up to 132 characters long, may contain any explanatory remark about the generation that you want stored in the VDE database.

The remark string is used as the title of the SCT note, if VDE was asked to enter the SCT note by the specification of the /SCT qualifier.

/SCT=filename

The **filename** parameter specifies the name of a file to be loaded into the SCT notes conference. This file contains the previously-edited text of an SCT entry for the SCT note associated with this replacement.

If this qualifier is used, the default notes conference for the stream—SCT-*stream-name*—is assumed.

If this qualifier is specified, the /REMARK string is used as the title of the SCT note.

For information on the proper creation and protection of an SCT notes conference, see the *Guide to Using VDE*.

Also see /FOLD and /OTHER.

/STREAM=stream-name

Specifies that generations for the development stream given by the **stream-name** parameter be modified. This qualifier thus affects the interpretation of generation numbers on the command’s **gen-spec** parameter. If this qualifier is omitted, VDE modifies generations for the default development stream.

EXAMPLES

1

```
VDE> MODIFY GENERATION [FACIL]MODNAM.C;5/REMARK="New remark string"
%VDE-I-GENMOD, information for generation [FACIL]MODNAM.C;5 modified in the
 database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example changes the remark string associated with generation 5 of module MODNAM.C in facility FACIL. The log messages confirm that the attributes of the module were modified and that the database transaction successfully completed.

2

```
VDE> MODIFY GENERATION MODNAM.MAR /OBSOLETE
%VDE-I-GENMOD, information for generation [RMS]MODNAM.MAR;6 modified in the
 database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example illustrates how to mark a module as obsolete in the default stream. The module will be dropped from all future streams descended from the default stream, but will remain accessible in the default stream.

MODIFY GROUP

Modifies the attributes and membership of a source group or build group in the VDE database. The attributes of a group that you can modify are the group's name and remark string. The membership of a group is the list of modules that are members of the group.

Requires CREGRP privilege.

FORMAT

MODIFY GROUP *group-name [= mod-name [, mod-name...]]*

PARAMETERS

group-name

The name of a group whose definition is to be modified. This group must be defined in the default development stream. The group name may not contain wildcard characters.

mod-name

The name of a module that should be added to or removed from the specified group. If the group is a source group, this module must be a source module. The module name consists of an optional facility name enclosed in square brackets, a module name, and an optional type name preceded by a period (for example, [FACNAM]MODNAM.MAR). If no facility name is specified, the default facility is assumed. If no type name is specified, all source modules (for a source group) or all modules (for a build group) with the given module name in the given facility are added to or removed from the group.

You can add or remove more than one module at a time by using wildcard characters in any of the three components of the module name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Those source modules (for a source group) or those modules (for a build group) whose names match the wildcard pattern are added to or deleted from the group.

If no module-name parameters are specified, the membership of the group is not changed.

DESCRIPTION

A group is always defined in a specific development stream. You can only refer to the group from within that stream. As a result, the MODIFY GROUP command only modifies the group definition in the default development stream.

To modify group definitions for more than one development stream, you must use the MODIFY GROUP command for each stream. For each stream, first enter the SET STREAM command to set the development stream then enter the MODIFY GROUP command to change the group in that stream.

To apply a qualifier to the command as a whole, place the qualifier immediately after the MODIFY GROUP keywords and before the **group-name** parameter. With this placement, qualifiers such as the /REMARK qualifier apply to the entire specified group or, as with such qualifiers as /ADD or /REMOVE, to all modules specified after the equal sign.

Only the /ADD and /REMOVE qualifiers can modify an individual module-name parameter. To add or remove an individual module, place the /ADD or /REMOVE qualifier immediately after the module-name parameter. The /ADD or /REMOVE qualifier placed after an individual module-name parameter overrides the command-level qualifier for that one parameter.

QUALIFIERS

/ADD (default)
/REMOVE

Controls whether the modules specified in the parameter list are added to or removed from the group. The /ADD qualifier adds the specified modules to the group and the /REMOVE qualifier deletes them from the group. To set the default for the command as a whole, specify the qualifier after the MODIFY GROUP keywords. To override that default for a single module, specify the qualifier after the **mod-name** parameter you choose to add or delete. If you do not specify one of these qualifiers, modules are added to the group.

Of /ADD and /REMOVE, /ADD is the default.

/BUILD
/SOURCE (default)

Specifies whether a source group or build group is modified. The /SOURCE qualifier causes VDE to modify the source group with the specified name. The /BUILD qualifier causes VDE to modify the build group with the specified name. If neither qualifier is specified, VDE modifies a source group.

Of /BUILD and /SOURCE, /SOURCE is the default.

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want the group definition modified. The /CONFIRM qualifier causes VDE to print a message for the group it is about to modify giving the name of the group and asking whether you want that group modified. If you answer YES (or Y), the group definition is modified. If you answer NO (or N), the group definition is not modified. The /NOCONFIRM qualifier causes VDE to modify the group definition without asking for confirmation.

/LOG (default)
/NOLOG

Controls whether log messages are printed after the group definition is modified. The /LOG qualifier causes such messages to be printed and the /NOLOG qualifier suppresses them. The messages indicate that the group has been modified and that the database transaction has successfully committed.

/NAME=group-name

Specifies a new name for the indicated group. After the group name is changed, the group must be referred to by the new name in all subsequent VDE commands. The old name is lost from the database. The new name can be up to 39 characters long and must follow normal VDE name syntax.

/REMARK="string"

Specifies a remark string to be stored for the specified group. This remark string replaces the remark specified with the CREATE GROUP command that created the group. The quoted remark string can be up to 132 characters long and may contain any explanatory remark about the group that you want to store in the VDE database.

EXAMPLES

1

```
VDE> MODIFY GROUP/REMARK="New remark string" GRPNAM
%VDE-I-GRPMOD, information for group GRPNAM modified in the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This command modifies the remark string associated with source group GRPNAM. The log messages confirm that the group definition was modified and that the database transaction successfully committed.

2

```
VDE> MODIFY GROUP/BUILD/NOLOG/NAME=NEWNAM OLDNAM
VDE> SHOW GROUP/BUILD
Build group NEWNAM
VDE>
```

In this example, the name of build group OLDNAM is changed to NEWNAM. The /NOLOG qualifier suppresses the log messages. A subsequent SHOW GROUP command shows the new name of the group.

3

```
VDE> MODIFY GROUP/SOURCE GRP1 = A, B.MAR/REMOVE
%VDE-I-GRPMOD, information for group GRP1 modified in the database
%VDE-I-GRPADD, module [CODE]A.PAS is added to group GRP1
%VDE-I-GRPREM, module [CODE]B.MAR is removed from group GRP1
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This command adds module A.PAS to the source group named GRP1 and removes module B.MAR from GRP1. No type name is specified for module A; hence all source modules named A in the default facility are added. As the log messages indicate, there is only one such module, A.PAS, in this example.

4

```
VDE> MODIFY GROUP/REMOVE GRP1 = A.PAS/ADD, B.MAR
%VDE-I-GRPMOD, information for group GRP1 modified in the database
%VDE-I-GRPADD, module [CODE]A.PAS is added to group GRP1
%VDE-I-GRPREM, module [CODE]B.MAR is removed from group GRP1
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This command also adds module A.PAS to group GRP1 and removes module B.MAR. Because the /REMOVE qualifier immediately follows the MODIFY GROUP keywords, all modules specified in the parameters are removed by default. This default applies to module B.MAR because it is not overridden. However, the /ADD qualifier specified for module A.PAS

overrides the default for that module and A.PAS becomes a member of the group.

MODIFY LIBRARY

Modifies information in the database header record for the current VDE library. This command can modify the library name, the library remark string, the standard authorized and default privileges for new users of the library, and various other library attributes. It can also enable or disable after-image journalling for the library's VDE database.

Requires MODLIB privilege.

FORMAT MODIFY LIBRARY

PARAMETERS

None.

QUALIFIERS

/ALLOW_DELETE
/NOALLOW_DELETE

Specifies the default value for the **delete** attribute for new modules, facilities, and streams. The /ALLOW_DELETE qualifier sets a library attribute that causes VDE to set the **delete** attribute by default when users create new such entities. As a result, users can delete these entities without first having to set the **delete** attribute with a separate MODIFY command. The /NOALLOW_DELETE qualifier clears this library attribute so that VDE clears the **delete** attribute by default for new modules, facilities, and streams. As a result, it is harder for a user to delete an entity by mistake; the entity's **delete** attribute must be set with a separate MODIFY command before the entity can be deleted with a DELETE command. When creating new modules, facilities, and streams, you can override this default setting with an explicit /DELETE or /NODELETE qualifier to the CREATE command.

Use the /ALLOW_DELETE qualifier if you expect to make frequent changes to your library, often deleting modules, facilities, and streams. Use the /NOALLOW_DELETE qualifier if you are more concerned about preventing accidental deletion of such entities.

/ASK_INFO_FILE
/NOASK_INFO_FILE

Specifies whether the REPLACE command should by default ask the user whether he or she wants to edit an information file when creating a queued replacement. The /ASK_INFO_FILE qualifier sets a library attribute that causes the REPLACE command to ask for an information file by default. The /NOASK_INFO_FILE qualifier clears this library attribute so that the REPLACE command does not ask for an information file by default. This library attribute only affects queued replacements (not immediate replacements), and it only affects the REPLACE command's default behavior; users can always use the command's /INFORMATION or /NOINFORMATION qualifier to explicitly request either behavior.

/ASK_REPL_CAUSE
/NOASK_REPL_CAUSE

Specifies whether the REPLACE command should by default ask the user for the ‘‘cause’’ of the replacement. The cause may be a fold operation, a Source Control Tracking (SCT) entry in a Notes conference, or some other cause. The /ASK_REPL_CAUSE qualifier sets a library attribute that causes the REPLACE command to query the user for the cause of the replacement. The /NOASK_REPL_CAUSE qualifier clears this library attribute so that the REPLACE command does not ask for this information.

/AUTH_PRIVILEGES=(priv [,priv...])

Specifies the standard authorized privileges of new users of the current library. However, any privileges specified with the CREATE USER command override these privileges. Because the privilege list you create replaces the authorized privilege list of the current library, you must specify all authorized privileges that you want new users of the library to have.

Each parameter to this qualifier names a single privilege. If only one privilege is specified, the parentheses may be omitted.

For a list of privileges, see Table 3–5 in SET PRIVILEGES.

/AUTO_ADD_USER[=rights-ident]
/NOAUTO_ADD_USER

Specifies whether new users should be added to the library database automatically when they first access the library. The /AUTO_ADD_USER qualifier specifies that new users should be added to the library database automatically when they first access the library. If the **rights-ident** parameter is specified, new users are only added to the library if they hold the OpenVMS rights identifier specified by the parameter. The /NOAUTO_ADD_USER qualifier specifies that new users not be added to the library database automatically. In this case, new users can only be added with explicit CREATE USER commands.

The owner information specified in the OpenVMS SYSUAF database will be used as the owner of any new VDE username created. VDE will ignore any leading numerics, leading hyphens, leading space characters, and leading tilde characters, listed in the SYSUAF owner field.

/AUTO_CONVERT
/NOAUTO_CONVERT

Specifies whether the library and its database are automatically converted to the new library format when a new version of VDE is used on the library. The /AUTO_CONVERT qualifier specifies that the library should be converted automatically the first time a user uses a new version of VDE on the present library and its database. /NOAUTO_CONVERT specifies that the library should not be converted automatically; instead a privileged user must explicitly convert the library with the CONVERT LIBRARY command. See the CREATE LIBRARY and the CONVERT LIBRARY commands.

/AUTO_MAIL
/NOAUTO_MAIL

Specifies whether VDE should automatically send mail notification to the user who has queued a replacement when that replacement is later performed. The /AUTO_MAIL qualifier sets a library attribute that causes the PERFORM REPLACEMENT command to send such mail notification. The /NOAUTO_MAIL qualifier clears this library attribute so that the PERFORM REPLACEMENT command does not send such notification.

/CMS_ELEM_HIST
/NOCMS_ELEM_HIST

Specifies whether VDE creates the CMS elements for new modules with default CMS history and notes strings. The /CMS_ELEM_HIST qualifier causes VDE to create the CMS elements for new modules with default history and notes strings and the /NOCMS_ELEM_HIST qualifier causes VDE to create CMS elements with no default history or notes strings. /NOCMS_ELEM_HIST is normally only used if you regularly fetch VDE modules using the CMS FETCH command and you want to make sure you do not get CMS history or notes by default.

/DEF_PRIVILEGES=(priv [,priv...])

Specifies the standard default privileges of new users of the current library. However, any privileges specified with the CREATE USER command override these privileges. Because the privilege list you create replaces the default privilege list of the current library, you must specify all default privileges that you want new users of the library to have. Each default privilege specified also becomes a standard authorized privilege for the library.

Each parameter to this qualifier names a single privilege. If only one privilege is specified, the parentheses may be omitted. For a list of privileges, see Table 3–5 in SET PRIVILEGES.

/HISTORY_NOTES
/NOHISTORY_NOTES

Specifies whether VDE's RESERVE command can produce CMS history or notes information in its output files. The /HISTORY_NOTES qualifier allows the RESERVE command to produce history and notes information and to accept the /HISTORY, /NOTES, and /POSITION qualifiers. The /NOHISTORY_NOTES qualifier causes the RESERVE command to never produce history and notes information (even when such information is the default for a module being reserved) and to not accept the /HISTORY, /NOTES, and /POSITION qualifiers. In this case, you must use the VDE FETCH command to get CMS history or notes information for a module. Disabling history and notes information in reserved files makes certain mistakes less likely where such information gets replaced into the VDE library as part of the module text.

/JOURNAL=dir-spec
/NOJOURNAL

Enables or disables after-image journalling for the VDE database. The /JOURNAL qualifier enables after-image journalling and specifies that a database after-image journal (.AIJ) file be created in the directory location given by the **dir-spec** parameter. When used with the RMU backup and

restore facilities, after-image journalling allows you to recover the up-to-date contents of your database after a disk failure. The specified directory should always reside on a different physical disk than any of your other database files so that the journal file cannot be damaged by the same disk failure as the database itself.

The /NOJOURNAL qualifier disables after-image journalling for the VDE database. However, Digital recommends that you always use journalling.

/LOCK
/UNLOCK

The /LOCK and /UNLOCK qualifiers provide a method by which RESERVE, REPLACE, UNRESERVE, and other similar commands can be disabled on a copy of a VDE library. This mechanism is not intended to completely lock-out all access to a VDE library.

Specification of /LOCK write-locks the library, and the corresponding /UNLOCK qualifier reverses the lock. The write-lock status is read only when the user initially accesses the library—once a user has connected to the library, the lock status is *not* updated.

A write-locked library can be used to prevent various user modifications to the contents of the library during library maintenance, or to maintain a shadow copy of a library.

Users holding the MODLIB privilege are allowed to override the library write-lock. An informational message will be displayed when the lock is overridden.

/LOG (default)
/NOLOG

Controls whether log messages are printed after the database header record of the library is modified. The /LOG qualifier causes such messages to be printed and /NOLOG suppresses them. The messages indicate that library attributes have been modified and that the database transaction has successfully committed.

/NAME=lib-name

Specifies a new name for the current library. The new library name can be up to 39 characters long and must follow normal VDE name syntax.

This value is available to procedures calling VDE via the VDE$LIBRARY_MNEMONIC name, and the database manager should set this value to match the values associated with this library that are specified in the VDE$LIBRARY_DEFAULTS_MNEMONICS logical name.

/REMARK="string"

Specifies a new remark string for the library. The quoted remark string, which can be up to 132 characters long, may contain any explanatory remark about the library that you want stored in the library database. This string replaces the remark string from the original CREATE LIBRARY command or from any previous MODIFY LIBRARY command.

This value is available to procedures calling VDE via the VDE$LIBRARY_LABEL name, and the database manager should set this value to match any values associated with this library that are specified in the VDE$LIBRARY_DEFAULTS_LABELS logical name.

/REPL_HISTORY
/NOREPL_HISTORY

Specifies whether VDE should maintain a history of all past source code replacements into the VDE library. The /REPL_HISTORY qualifier specifies that such history information be maintained in the VDE database. The /NOREPL_HISTORY qualifier specifies that such information not be maintained. When you maintain replacement history, you can display past replacements using the /HISTORY qualifier to the SHOW REPLACEMENT command.

/SEND_MAIL
/NOSEND_MAIL

Specifies whether VDE should send mail notification messages for those VDE operations that normally send such messages. The /SEND_MAIL qualifier sets a library attribute that enables VDE to send such mail messages. The /NOSEND_MAIL qualifier clears that attribute, thus suppressing all VDE mail messages. The /NOSEND_MAIL qualifier is useful when you want to test or play with a private VDE library without sending notification messages to other people. However, for normal production use, you should always enable mail messages.

/SHOW_ARCH_NAME
/NOSHOW_ARCH_NAME

Specifies whether the architecture name is displayed as part of every module name in VDE output. The /SHOW_ARCH_NAME qualifier sets a library attribute that causes VDE to always include the architecture name in every module name it displays. This attribute also causes the SHOW DEFAULTS command to display the default architecture. The /NOSHOW_ARCH_NAME qualifier clears this library attribute so that VDE omits the architecture names when displaying module names. You normally only want to see the architecture name if you have defined multiple architectures for your VDE library.

This qualifier is not yet implemented.

/SHOW_GEN_EXPR
/NOSHOW_GEN_EXPR

Specifies whether the expected replace-time CMS generation expression for each module is displayed by the RESERVE command. The /SHOW_GEN_EXPR qualifier sets a library attribute that causes the RESERVE command to compute and display the CMS generation expression that the corresponding REPLACE command is most likely to use for the new module generation. You can use this value as an ‘‘audit trail identifier’’ in your code if your project uses this convention. This library attribute also causes the SHOW RESERVATION command to display the expected CMS generation expression and it causes the REPLACE or PERFORM REPLACEMENT command to print an informational message if the actual CMS generation expression for the replaced generation turns out to be different than the expected one. The /NOSHOW_GEN_EXPR qualifier disables this library attribute.

Please note that VDE cannot guarantee that the expected CMS generation expression will actually be used. It is not possible to determine at RESERVE time whether a CMS variant letter should be used when the module is eventually replaced; that decision can only be made at REPLACE time and depends on propagation decisions and stream

successor relationships at that time. However, the expected CMS generation expression is generally the one that is actually used.

/STATISTICS

Enables statistics collection on the specified library. This will cause various extra questions to be asked during replacement and related operations. This includes the addition of questions around the reason for the change, the number of defects fixed, and the particular project a change is associated with.

/WILD_INS_GEN
/NOWILD_INS_GEN

Specifies whether wildcard characters are allowed in the parameters to the INSERT GENERATION command when the /GENERATION qualifier is specified. The /WILD_INS_GEN qualifier sets a library attribute that causes VDE to allow wildcard characters in such parameters while the /NOWILD_INS_GEN qualifier clears this attribute. When the /GENERATION qualifier to INSERT GENERATION specifies a specific CMS generation expression, it is unlikely that this generation expression could apply to more than one module. As a result, it is unlikely that the use of wildcard characters in the module-name parameters will give the result the user intended. To prevent mistakes, VDE therefore does not allow such wildcard usage by default. However, there are some situations when such usage might be correct (for example, if the specified CMS generation expression is 1). For such situations, you can enable the use of wildcard characters with the /WILD_INS_GEN qualifier.

EXAMPLES

1
```
VDE> MODIFY LIBRARY/DEF_PRIVILEGES=(BUILD,CREMOD)
%VDE-I-LIBMOD, library attributes modified in database
%VDE-I-COMMIT, database transaction has successfully committed
```

In this example, the standard default privileges for new users of the library are modified to consist of the BUILD and CREMOD privileges. The log messages confirm that the library has been modified and that the database transaction successfully completed.

2
```
VDE> MODIFY LIBRARY/REMARK="New remark string"
%VDE-I-LIBMOD, library attributes modified in database
%VDE-I-COMMIT, database transaction has successfully committed
```

This example modifies the remark string for the current library.

MODIFY MODULE

Modifies the attributes of modules in the VDE database.

Requires CREMOD privilege.

FORMAT

MODIFY MODULE *mod-name [, mod-name...]*

PARAMETER

mod-name

The name of an existing module in the current VDE library whose attributes are to be modified. The module name must consist of a module name and a module type separated by a period (as in MODNAM.FOR) and may optionally be preceded by a facility name enclosed in square brackets (as in [ACCTNG]MODNAM.FOR where ACCTNG is the facility name). If no facility name is specified, the module is assumed to belong to the current default facility.

You can modify more than one module by using wildcard characters in any of the three components of the module name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. VDE modifies the attributes of those modules whose names match the wildcard pattern.

DESCRIPTION

The MODIFY MODULE command allows you to modify various attributes associated with a module. The MODIFY MODULE commands apply to attributes that span a number of module generations.

For information on changing the attributes specific to a particular module generation, see MODIFY GENERATION. The MODIFY GENERATION/MAIN_STREAM, /OBSOLETE, and /REMARK qualifiers allow you to mark a module as the main line of descent, to mark a module as obsolete, or to change the remark associated with a particular module generation.

QUALIFIERS

/BINARY
/NOBINARY

Specifies whether the source module is a binary file or not. The /BINARY qualifier specifies that the source module is a binary (non-text) file. Binary source files cannot be reviewed or differenced with the REVIEW REPLACEMENT command because they do not consist of ordinary ASCII text. The /NOBINARY qualifier specifies that the source module is an ordinary ASCII text file. These qualifiers are only meaningful for source modules.

Large files that undergo extensive changes at frequent intervals, or are the output of a language or application, may not be suited for storage as standard delta files as the file delta is never used, occupies a considerable amount of disk space, and grows rapidly. See the /MARKER qualifier. Executable images, object modules, Postscript files, and BACKUP savesets are all example of ''source'' modules for which the marker-file mechanism may be appropriate.

/CONCURRENT
/NOCONCURRENT

Specifies whether the module can be reserved concurrently by more than one user. The /CONCURRENT qualifier allows the module to be reserved concurrently unless a user overrides this default with for a specific reservation. The /NOCONCURRENT qualifier specifies that the module cannot be reserved concurrently.

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want the attributes of each module modified. The /CONFIRM qualifier causes VDE to print a message for each module giving the name of the module and asking whether you want to modify the module. If you answer YES (or Y), the module's attributes are modified. If you answer NO (or N), the attributes are not modified. The /NOCONFIRM qualifier causes VDE to modify the attributes of each module without asking for confirmation.

/DELETE
/NODELETE (default)

Sets or clears the **delete** attribute for the module. This attribute allows the module to be deleted with the DELETE MODULE command. The /DELETE qualifier sets the delete attribute so that the module can be deleted. The /NODELETE qualifier clears the delete attribute so that the module cannot be deleted without first resetting the attribute.

/HISTORY[="string"]
/NOHISTORY[="string"]

Specifies the CMS history string to be associated with the module and whether history information should be included in source files when modules are retrieved with the FETCH or RESERVE command. History information gives a historical record of previous changes to the module.

The /HISTORY qualifier specifies that history information should be included in source files by default when such files are retrieved with the FETCH or RESERVE command. The /NOHISTORY qualifier specifies that history information should be omitted. The **string** parameter to either qualifier specifies the default history string if the user specifies that history information should be included in a retrieved file. (If you specify the /NOHISTORY qualifier on the MODIFY MODULE command, you must use the /HISTORY qualifier on the FETCH or RESERVE command to use the default history string.) If you do not specify a **string** parameter, VDE supplies a default history string that depends on the module type.

See the CMS documentation for a description of the CMS history string and its format.

/LOG (default)
/NOLOG

Controls whether log messages are printed after the attributes of modules are modified. The /LOG qualifier causes such messages to be printed and /NOLOG suppresses them. The messages indicate that the modules have been modified and that the database transaction has successfully committed.

/MARKER
/NOMARKER

Specifies whether the source module uses a marker file or not. The /MARKER qualifier specifies that the source module uses a marker file. When a source module uses a marker file, the module is stored as an ordinary file in one of the delta-file directories. A small text file, called the marker file, contains the full file name of that source file, and this marker file is stored in the module's delta file (CMS element). In short, the delta file contains a small text file that points to the module's actual source file. The marker-file mechanism is normally only used for large or frequently changed binary source modules, for which the delta-file mechanism may give poor performance or use excessive disk space.

The /NOMARKER qualifier specifies that the source module does not use a marker file and that its text is stored directly in the delta file (CMS element) for the module. Ordinary ASCII text modules should not use marker files.

Also see the /[NO]BINARY qualifier.

/NOTES["string"]
/NONOTES["string"]

Specifies the CMS notes string to be associated with the module and whether CMS notes should be included in source files when modules are retrieved with the FETCH or RESERVE command. CMS notes indicate in what module generation each source line was created.

The /NOTES qualifier specifies that notes information should be included in source files by default when such files are retrieved with the FETCH or RESERVE command. The /NONOTES qualifier specifies that notes information should be omitted. The **string** parameter to either qualifier specifies the default notes string if the user specifies that notes information should be included in a retrieved file. (If you specify the /NONOTES qualifier on the MODIFY MODULE command, you must use the /NOTES qualifier on the FETCH or RESERVE command to use the default notes string.) If you do not specify a **string** parameter, VDE supplies a default notes string that depends on the module type.

See the CMS documentation for a description of the CMS notes string and its format.

/NOTIFY=(username [,username...])
/NONOTIFY[=(username [,username...])]

The /NOTIFY qualifier specifies a list of users to notify of all changes to the specified module. VDE appends these users to the module's existing notification list. Each **username** parameter gives the OpenVMS username or mail address of a user who should be notified by mail message each time the specified module is replaced into the VDE library.

Each **username** parameter can specify the username of a user already recorded in the VDE database, a OpenVMS mail address with one or more node names (such as NODE::SMITH), a mail forwarding alias, or a defined logical name bound to a list of users to notify. If you specify a logical name, keep in mind that the logical name must be defined when it is later used to send notification messages. It should thus be a group or system logical name or a logical name that is in some other way defined for all users of the VDE library.

The /NONOTIFY qualifier specifies that a list of users already on the module's notification list be removed from that notification list. Each **username** parameter gives the OpenVMS username or mail address of a user to remove from the notification list. If you omit the **username** parameters, VDE removes the entire notification list for the module.

If you specify both qualifiers, VDE removes the users specified with the /NONOTIFY qualifier from the notification list before adding the users specified with the /NOTIFY qualifier.

If you specify only one **username** parameter with either qualifier, you may omit the surrounding parentheses.

/POSITION=column-number

Specifies the default column number in which CMS notes are to be placed when this source module is retrieved with a FETCH or RESERVE command. The column number must be an integer in the range 1 to 511. If this qualifier is omitted, the default column number is 81. This qualifier has no effect unless notes are enabled with the /NOTES qualifier on the CREATE or MODIFY MODULE command or on the FETCH or RESERVE command.

See the CMS documentation for a description of CMS notes and how notes are formatted at the ends of source lines.

/PROPAGATE
/NOPROPAGATE

Specifies whether automatic change propagation is allowed for this module when it is reserved and replaced into the library. /PROPAGATE allows automatic change propagation and /NOPROPAGATE disallows change propagation. If change propagation is disallowed, VDE automatically inhibits change propagation when the module is reserved or replaced as if the /NOPROPAGATE qualifier had been specified for the RESERVE or REPLACE command.

/REMARK="string"

Specifies a remark string to be stored for the module. This string replaces the remark string from the original CREATE MODULE command or from any previous MODIFY MODULE command. The quoted remark string, which can be up to 132 characters long, may contain any explanatory remark about the module that you want stored in the VDE database. Typically the remark describes the function or contents of the module.

This qualifier alters the module-level remark string. To change the remark on a particular module generation, see the documentation on the MODIFY GENERATION/REMARK command in MODIFY GENERATION.

/REVIEWER=(username [,username...])
/NOREVIEWER[=(username [,username...])]

The /REVIEWER qualifier specifies a list of reviewers for all changes to the specified module. VDE appends these reviewers to the module's existing reviewer list. Each **username** parameter gives the OpenVMS username of a user who should review all changes to the module. The user must already be recorded in the VDE database. When the module is replaced into the VDE library, VDE notifies the user of the change by mail message. If the replacement is queued, the user should then review the queued replacement before it is performed. If the replacement is immediate, VDE notifies the user of the new module generation that was created, and the user may then inspect that generation.

The /NOREVIEWER qualifier specifies that a list of existing reviewers for the specified module be removed from the reviewer list. Each **username** parameter gives the OpenVMS username of a user who should be removed from the reviewer list. If you omit the **username** parameters, VDE removes the entire reviewer list for the module.

If you specify both qualifiers, VDE removes the users specified with the /NOREVIEWER qualifier from the reviewer list before adding the users specified with the /REVIEWER qualifier.

If you specify only one **username** parameter with either qualifier, you may omit the surrounding parentheses.

EXAMPLES

1

```
VDE> MODIFY MODULE MODNAM.C/REMARK="New Comment"
%VDE-I-MODMOD, information for module [CODE]NEWNAM.C modified in the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example changes the remark present on the module MODNAM.C. Since no facility is specified, the module is assumed to belong to the default facility, facility CODE in this case. The log messages confirm that the attributes of the module were modified and that the database transaction successfully completed.

2

```
VDE> MODIFY MODULE/NOLOG [TAPE]DEVDRV.MAR -
_VDE>                    /REMARK="Old tape device driver"
VDE> MODIFY GENERATION [TAPE]DEVDRV.MAR -
_VDE>                  /NOLOG /OBSOLETE
VDE>
```

In this example, the user changes the remark string for module DEVDRV.MAR in facility TAPE, and then uses the MODIFY GENERATION command to mark the module as obsolete. The /NOLOG qualifier is specified on both commands; the qualifier suppresses the display of messages from either.

MODIFY REPLACEMENT

Modifies the attributes of one or more queued replacements in the VDE database. The attributes of a replacement you can modify are its name, remark string, and reviewer list.

Requires RESREP privilege to modify your own replacement. Requires PERFREP privilege to modify another user's replacement.

FORMAT

MODIFY REPLACEMENT *rep-name [, rep-name...]*

PARAMETER

rep-name

The name of a queued replacement whose attributes are to be modified. You can modify more than one replacement at a time by using wildcard characters in the replacement name. The percent sign (%) in a replacement name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. VDE modifies the attributes of those replacements whose names match the wildcard pattern.

Do not specify the **rep-name** parameter if you use the /IDENTIFICATION qualifier.

QUALIFIERS

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each queued replacement modified. The /CONFIRM qualifier causes VDE to print a message for each replacement asking whether you want that replacement modified. If you answer YES (or Y), the replacement is modified. If you answer NO (or N), the replacement is not modified. If you enter a replacement name with a wildcard specification, VDE asks you to confirm the modification of each replacement whose name matches the wildcard specification. The /NOCONFIRM qualifier causes VDE to modify the specified replacements without asking for confirmation.

/IDENTIFICATION=id-range

Specifies that the replacements with the unique Replacement Ids given by the **id-range** parameter be modified. Do not specify the **rep-name** parameter when you use this qualifier.

VDE assigns a unique Replacement Id to each immediate or queued replacement into the VDE library. VDE assigns Replacement Ids in chronological order so that the first replacement into the library has Id value 1, the next has Id value 2, and so on. When you use this qualifier, all replacements that have Id values in the range given by the **id-range** parameter are modified.

The **id-range** parameter can be a single integer value, a range of integers, or a wildcard character. If the parameter is a single integer (for example, /IDENT=5), VDE modifies the replacement with that Replacement Id. If the parameter is a range of integers (specified as two integers separated by a colon as in /IDENT=3:5), VDE modifies the replacements with Ids in that range. Finally, if the **id-range** parameter is the asterisk wildcard character (as in /IDENT=*), VDE modifies all replacements.

If you specify a zero or negative number in the **id-range** parameter, that number is relative to the largest Replacement Id value for the VDE library (the latest replacement). For example, if the latest replacement has Replacement Id 250, then /IDENT=0 is equivalent to /IDENT=250 and /IDENT=–1 is equivalent to /IDENT=249.

/INFORMATION[=file-spec]

Specifies that a new information file should be attached to the queued replacement or that the existing information file should be edited. The **file-spec** parameter gives the OpenVMS file specification of a new information file for the replacement. If specified, this file is copied to the replacement's staging area and then supercedes any previous information file for the replacement. If this parameter is omitted, VDE asks you whether you want to edit the information file. If you answer YES, VDE invokes your preferred editor to edit the replacement's information file. If an information file already exists, the editor edits that file; otherwise it creates a new information file. The information file is used to pass information about the replacement to reviewers and others.

To edit an information file through VDE, you should first declare your preferred editor with the SET EDITOR command. If you do not declare an editor, VDE invoked the system default text editor.

/LOG (default)
/NOLOG

Controls whether log messages are printed after the replacements are modified. The /LOG qualifier causes such messages to be printed and /NOLOG suppresses them. The messages indicate that the replacements have been modified and that the database transaction has successfully committed.

/NAME=rep-name

Specifies a new name for the queued replacement. The new replacement name can be up to 39 characters long and must follow normal VDE name syntax. After the replacement name has been changed, you must refer to the replacement by its new name in all subsequent VDE commands. The old name is lost from the database.

/REMARK="string"

Specifies a new remark string for the replacement. This string replaces the remark string from the original REPLACE or CREATE REPLACEMENT command or from any previous MODIFY REPLACEMENT command. The quoted remark string, which can be up to 132 characters long, may contain any explanatory remark about the replacement that you want stored in the VDE database.

/REVIEWER=(username [,username...])
/NOREVIEWER[=(username [,username...])]

The /REVIEWER qualifier specifies a list of reviewers for the replacement. VDE appends these reviewers to the replacement's existing reviewer list. Each **username** parameter gives the OpenVMS username of a user who should review all code included in the replacement. The user must already be recorded in the VDE database. VDE notifies the user by mail message that he or she should review the replacement.

The /NOREVIEWER qualifier specifies that a list of existing reviewers for the replacement be removed from the reviewer list. Each **username** parameter gives the OpenVMS username of a user who should be removed from the reviewer list. If you omit the **username** parameters, VDE removes the entire reviewer list for the replacement.

If you specify both qualifiers, VDE removes the users specified with the /NOREVIEWER qualifier from the reviewer list before adding the users specified with the /REVIEWER qualifier.

If you specify only one **username** parameter with either qualifier, you may omit the surrounding parentheses.

EXAMPLES

1
```
VDE> MODIFY REPLACEMENT SMITH-2/REMARK="New remark string"
%VDE-I-REPMOD, information for replacement SMITH-2 modified in the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example modifies replacement SMITH–2 to give it a new remark string. The log messages show that the replacement has successfully been modified.

MODIFY RESERVATION

Modifies the attributes of module reservations in the VDE database.

Requires RESREP privilege.

FORMAT

MODIFY RESERVATION *mod-name [,mod-name...]*

PARAMETER

mod-name

Specifies a reserved source module whose reservation attributes are to be modified. The module name consists of an optional facility name enclosed in square brackets, a module name, and an optional type name preceded by a period (such as [FACIL]MOD1.MAR). If the facility name is omitted, the module is assumed to belong to the default facility. If the type name is omitted, the reservation attributes of all source modules with the given module name in the given facility are modified.

You can modify the reservation attributes of more than one module using wildcard characters in any component of the module name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. The reservation attributes of the source modules whose names match the wildcard pattern are modified.

If you use the /SESSION qualifier, those reservations that are members of the specified reservation session are modified. In this case, you must omit the **mod-name** parameters from the command.

DESCRIPTION

When you reserve a module from the VDE library, the RESERVE command records the reservation in the VDE database. The MODIFY RESERVATION command lets you modify various attributes of one or more such reservations. The command modifies only reservations that cover the default development stream or the stream specified with the /STREAM qualifier.

If you have more than one reservation of a module, you must specify the exact reservation to be modified using the /IDENTIFICATION qualifier. Use the SHOW RESERVATION command to determine the identification of each reservation.

To modify another user's reservation, use the /USERNAME qualifier to specify the OpenVMS username of that other user. You must have the USERNAME privilege to use the /USERNAME qualifier.

QUALIFIERS

/CONCURRENT

Specifies that the module can be reserved by another user while you have it reserved. This qualifier changes the concurrent-reservation attribute of the reservation.

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each reservation modified. The /CONFIRM qualifier causes VDE to print a message for each module that you specify asking whether you want to modify its reservation. If you answer YES (or Y), the reservation is modified. If you answer NO (or N), it is not modified. The /NOCONFIRM qualifier causes VDE to modify each specified reservation without asking for confirmation.

/FOLD[=fold-ident]
/NOFOLD

The /FOLD qualifier specifies a fold record to be cancelled when this module reservation is eventually replaced into the library. The **fold-ident** parameter gives the fold identifier of a fold record for the module being reserved and for the stream in which the reservation is done. When you later replace the module and VDE performs the replacement, VDE will cancel this fold record from the VDE database. You can omit the **fold-ident** parameter if there is only one fold record for the specified module and stream. You can use the /FOLD qualifier if you are modifying the module to satisfy an existing fold record. You can also specify the /FOLD qualifier with the REPLACE command to achieve the same effect.

The /NOFOLD qualifier specifies that no fold record be cancelled when this module reservation is replaced into the library. This qualifier cancels the effect of a /FOLD qualifier on a previous MODIFY RESERVATION command or on the original RESERVE command.

/IDENTIFICATION=res-ident

Specifies the reservation to be modified. This qualifier is required when you have multiple reservations of the same module in the default development stream. The **res-ident** parameter is the reservation identifier of the reservation to be modified. The reservation identifier is the identifier you specified when you reserved the module, usually a small integer value. Use the SHOW RESERVATIONS command to display the reservation identifier of each reservation.

/LOG (default)
/NOLOG

Controls whether log messages are printed after each reservation is modified. The /LOG qualifier causes the messages to be printed and the /NOLOG qualifier suppresses them. The messages indicate that each reservation has been modified and that the database transaction has successfully committed.

/NEW_SESSION=session-name

Specifies that VDE should add the module reservation to the reservation session given by the **session-name** parameter. If no session by that name exists, VDE creates a new reservation session by that name and adds the reservation to it. If the reservation already belongs to an existing session, it is removed from that session and added to the new session.

/NOCONCURRENT

Specifies that the module cannot be reserved by another user while you have it reserved. You must replace or unreserve the module before others can reserve it. This qualifier changes the concurrent-reservation attribute of the reservation.

/PROPAGATE=stream-name
/NOPROPAGATE

This qualifier controls the default module propogation behaviour of the subsequent REPLACE command, and it controls the reservation ‘‘shadow’’ for this particular reservation.

When specified, the /PROPAGATE qualifier causes VDE to propagate the changed module(s) to the stream specified by the **stream-name** parameter, but *not* to any successors of that stream—change propagation thus stops at the specified stream.

The /NOPROPAGATE qualifier causes VDE to disable all propagate of the change beyond the current stream (the default stream or the stream specified with the /STREAM qualifier).

If you omit both qualifiers, the REPLACE command will propagate the changed modules to all successors of the current stream.

Because these qualifiers determine what streams a subsequent REPLACE command will affect, they also determine what streams—the ‘‘shadow’’—the new module reservations cover. A reservation that covers fewer streams allows other users to reserve the same module in other streams with less likelihood of conflict.

/REMARK="string"

Specifies a new remark string to be associated with the module reservation. This qualifier changes the remark attribute and the previous reservation remark is lost from the database. The remark is enclosed in quotation marks.

/SESSION=session-name

Sessions are used to logically group a set of module reservations together, typically to group all modules related to a particular source code alteration or enhancement together. It allows all component modules reserved to be treated as a single entity for subsequent replacement operations. A session also allows additional modules to be reserved and incorporated into an existing session at a later time.

If this qualifier is specified, the modules referenced by the **session-name** session are the target; do not specify the **mod-name** module name parameters.

Sessions can be manipulated via the REPLACE, RESERVE, UNRESERVE, MODIFY SESSION, MODIFY RESERVATION, CREATE MODULE, and CANCEL SESSION commands. And modules created by CREATE MODULE (on a queued-replacement stream) and reserved via RESERVE can be combined into the same session.

/STREAM=stream-name

Specifies that reservations for the development stream given by the **stream-name** parameter be modified. If this qualifier is omitted, reservations for the default development stream are modified.

/USERNAME=username

Specifies that reservations made by another user are to be modified. The **username** parameter is the OpenVMS username of the other user. You must have the USERNAME privilege to use this qualifier.

EXAMPLES

1

```
VDE> MODIFY RESERVATION FOO /REMARK="Fix QARS 32 and 33"
%VDE-I-RESMOD, reservation for module [FACIL]FOO.MAR modified in the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE> SHOW RESERVATION FOO/FULL
Reservation 1 of [FACIL]FOO.MAR
   Reservation created on 29-DEC-1989 13:09:47.27
   Reservation created by user SMITH (Jane Smith)
   Generation 29 (27B2) reserved from stream V5.4
   Expected generation expression at replacement is 27B3
   Concurrent reservations are allowed
   Remark: Fix QARS 32 and 33

VDE>
```

This example modifies the reservation remarks of all source modules in the default facility with the name FOO. In this case, only module [FACIL]FOO.MAR matches the module name, and its reservation remark is changed. The SHOW RESERVATION command confirms that the reservation now has the new remark.

2

```
VDE> MODIFY RESERVATION/NOCONCURRENT [FACIL]MOD1.FOR
%VDE-I-RESMOD, reservation for module [FACIL]MOD1.FOR modified in the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE> SHOW RESERVATION/FULL MOD1.FOR
Reservation 1 of [FACIL]MOD1.FOR
   Reservation created on 29-DEC-1989 13:18:34.75
   Reservation created by user SMITH (Jane Smith)
   Generation 3 (3) reserved from stream MAIN
   Expected generation expression at replacement is 4
   No concurrent reservations are allowed
   Remark: Add new look-up algorithm

VDE>
```

The MODIFY RESERVATION command specifies that concurrent reservations are not allowed for module MOD1.FOR in facility FACIL. The output from the subsequent SHOW RESERVATION command shows that no concurrent reservations are allowed.

3

```
VDE> MODIFY RESERVATION MOD1.FOR/IDENT=2/REMARK="Fix uninitialized pointer"
%VDE-I-RESMOD, reservation for module [FACIL]MOD1.FOR modified in the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE> SHOW RESERVATION MOD1.FOR /FULL
Reservation 1 of [FACIL]MOD1.FOR
   Reservation created on 24-MAR-1989 11:24:48.77
   Reservation created by user SMITH (Jane Smith)
   Generation 1 (1) reserved from stream MAIN
   Expected generation expression at replacement is 2
   Concurrent reservations are allowed
   Remark: Fix access violation

Reservation 2 of [FACIL]MOD1.FOR
   Reservation created on 24-MAR-1989 11:25:36.85
   Reservation created by user SMITH (Jane Smith)
   Generation 1 (1) reserved from stream MAIN
   Expected generation expression at replacement is 2
   Concurrent reservations are allowed
   Remark: Fix uninitialized pointer

VDE>
```

This example modifies one of two concurrent reservations of the same module. Because there are concurrent reservations, the MODIFY

RESERVATION command must have the /IDENTIFICATION qualifier to specify which reservation to modify. In this case, the command modifies the remark for reservation 2 of [FACIL]MOD1.FOR. The SHOW RESERVATION command confirms that reservation 2 now has the new remark.

MODIFY SCRIPT

Modifies the remark string associated with a VDE script in the VDE database.

Requires CRESCRP privilege.

FORMAT MODIFY SCRIPT

DESCRIPTION

The MODIFY SCRIPT command modifies the remark string for a specified VDE script in the VDE database. VDE only modifies the script of the specified type that belongs to the default development stream.

You must specify a script-type qualifier to select the script you want modified. See the description of the CREATE SCRIPT command for an explanation of the different types of scripts and how scripts are selected using script-type qualifiers. You must also specify the script segment number unless the default value applies.

Keywords used within scripts are documented in Script Keywords, while script functions are in Script Functions, and script types are in Script Types. For other script-related commands, also see CREATE SCRIPT, DELETE SCRIPT, EXTRACT SCRIPT, INVOKE SCRIPT, and SHOW SCRIPT.

QUALIFIERS

/BUILD

Specifies that the default stream's BUILD script be modified.

/COMPILE=mod-name

Specifies that the default stream's COMPILE script with the specified **mod-name** specification be modified.

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each script modified. The /CONFIRM qualifier causes VDE to print a message for each script asking whether you want that script modified. If you answer YES (or Y), the script's attributes are modified in the database. If you answer NO (or N), the script is not modified. The /NOCONFIRM qualifier causes VDE to modify the specified scripts without asking for confirmation.

/COPY=mod-name

Specifies that the default stream's COPY script with the specified **mod-name** specification be modified.

/CREATE_FACILITY

Specifies that the script that VDE invokes when a new facility is created should be modified.

/DELETE_GENERATION=mod-name
Specifies that the default stream's DELETE_GENERATION script with the specified **mod-name** specification be modified.

/DIFFERENCES=mod-name
Specifies that the default stream's DIFFERENCES script with the specified **mod-name** specification be modified.

/FETCH=mod-name
Specifies that the default stream's FETCH script with the specified **mod-name** specification be modified.

/INSERT_GENERATION=mod-name
Specifies that the default stream's INSERT_GENERATION script with the specified **mod-name** specification be modified.

/LINK=mod-name
Specifies that the default stream's LINK script with the specified **mod-name** specification be modified.

/LOG (default)
/NOLOG
Controls whether log messages are printed after the script is modified. The /LOG qualifier causes such messages to be printed and the /NOLOG qualifier suppresses them. These messages indicate that the script has been modified and that the database transaction has committed successfully.

/NEW_STREAM=fac-name
Specifies that the default stream's NEW_STREAM script with the specified **fac-name** specification be modified.

/NOKEEP=mod-name
Specifies that the default stream's NOKEEP script with the specified **mod-name** specification be modified.

/REMARK="string"
Specifies a new remark string for the script. This string replaces the remark string from the original CREATE SCRIPT command or from a previous MODIFY SCRIPT command. The quoted remark string, which can be up to 132 characters long, may contain any explanatory remark about the script that you want stored in the VDE database.

/REPLACE=mod-name
Specifies that the default stream's REPLACE script with the specified **mod-name** specification be modified.

/RESERVE=mod-name
Specifies that the default stream's RESERVE script with the specified **mod-name** specification be modified.

/SEGMENT=seg-num
Specifies the segment number of the script to modify. The **seg-num** parameter specifies the integer segment number. If you omit this qualifier, VDE modifies the script with segment number 10.

/SET_FACILITY=fac-name
Specifies that the default stream's SET_FACILITY script with the specified **fac-name** specification be modified.

/SET_STREAM
Specifies that the default stream's SET_STREAM script be modified.

/STAGING=mod-name
Specifies that the default stream's STAGING script with the specified **mod-name** specification be modified.

/UNRESERVE=mod-name
Specifies that the default stream's UNRESERVE script with the specified **mod-name** specification be modified.

EXAMPLES

1
```
VDE> MODIFY SCRIPT/COMPILE=*.C/REMARK="New remark string"
%VDE-I-SCRPMOD, script information modified in the database
 script to compile module [CODE]*.C (segment 10)
%VDE-I-COMMIT, database transaction has successfully committed
VDE> SHOW SCRIPT/FULL/COMPILE=[CODE]*.C
Script to compile module [CODE]*.C (segment 10)
   Script added to database on  9-DEC-1989 20:34:25.44
   Script added to database by user JONES (John Jones)
   Remark: New remark string

VDE>
```

This example modifies the script to compile all C modules in facility CODE to have a new remark string. The log messages show that the script was successfully modified. The SHOW SCRIPT/FULL command shows that the database contains the new remark string.

MODIFY SESSION

Modifies the attributes of one or more reservation sessions in the VDE database. The attributes of a session you can modify are its name and remark string.

Requires RESREP privilege to modify your own session. Requires PERFREP privilege to modify another user's session.

FORMAT

MODIFY SESSION *session-name [, session-name...]*

PARAMETER

session-name

The name of a reservation session whose attributes are to be modified. You can modify more than one session at a time by using wildcard characters in the session name. The percent sign (%) in a session name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. VDE modifies the attributes of those sessions whose names match the wildcard pattern.

QUALIFIERS

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each reservation session modified. The /CONFIRM qualifier causes VDE to print a message for each session asking whether you want that session modified. If you answer YES (or Y), the session is modified. If you answer NO (or N), the session is not modified. If you enter a session name with a wildcard specification, VDE asks you to confirm the modification of each session whose name matches the wildcard specification. The /NOCONFIRM qualifier causes VDE to modify the specified sessions without asking for confirmation.

/LOG (default)
/NOLOG

Controls whether log messages are printed after the sessions are modified. The /LOG qualifier causes such messages to be printed and /NOLOG suppresses them. The messages indicate that the sessions have been modified and that the database transaction has successfully committed.

/NAME=session-name

Specifies a new name for the reservation session. The new session name can be up to 39 characters long and must follow normal VDE name syntax. After the session name has been changed, you must refer to the session by its new name in all subsequent VDE commands. The old name is lost from the database.

/REMARK="string"

Specifies a new remark string for the session. This string replaces the remark string from the original RESERVE or CREATE SESSION command or from any previous MODIFY SESSION command. The quoted

remark string, which can be up to 132 characters long, may contain any explanatory remark about the session that you want stored in the VDE database.

EXAMPLES

1

```
VDE> MODIFY SESSION SMITH-2/REMARK="New remark string"
%VDE-I-SESMOD, information for session SMITH-2 modified in the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example modifies session SMITH–2 to give it a new remark string. The log messages show that the session has successfully been modified.

MODIFY STEP

Modifies a build-step definition in the VDE database. This command lets you add or remove individual input and output modules to the build-dependency information of a compile or link step.

Requires CRESTEP privilege.

FORMAT

MODIFY STEP *step-mod-name = mod-name [, mod-name...]*

PARAMETERS

step-mod-name

The name of the module that is compiled or linked by the build step. This name consists of a module name and a type name separated by a period (such as MOD1.PAS) and optionally may include a facility name enclosed in square brackets (such as [FACNAM]MOD1.PAS). The module name may not contain wildcard characters. If no facility name is specified, the default facility is assumed. This module must already exist in the database and it must already have build-dependency information.

mod-name

The name of a module that you want to add or remove as an input or output for the build step. This name consists of a module name and a type name (for example, MOD1.OBJ) and may include a facility name enclosed in square brackets (for example, [FACNAM]MOD1.OBJ). If no facility name is specified, the default facility is assumed.

You can add or remove many input or output modules by specifying wildcard characters in any component of the module name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Those modules whose names match the wildcard pattern are added to or removed from the build step.

If the output module you specify does not already exist in the database, the MODIFY STEP command adds it to the database. Input modules, however, must already be defined in the database. Also, you cannot specify a source module as an output from a build step.

DESCRIPTION

To be able to perform minimal system builds, VDE records all inputs and outputs for each build step. This information is known as ‘‘build dependency information.’’ Normally, VDE collects build dependency information automatically when you perform system builds. For some build steps, however, this is not possible and you must use the CREATE STEP command to enter the build dependency information into the VDE database.

MODIFY STEP

The MODIFY STEP command is normally used to edit build step definitions created with the CREATE STEP command. The MODIFY STEP command allows you to change build dependency information in the VDE database by adding or removing individual input and output modules.

Build dependency information is always created in the default development stream. To modify a build step definition for use in more than one development stream, you must use the MODIFY STEP command for each stream. For each stream, first enter the SET STREAM command to set the development stream, then enter the MODIFY STEP command to modify the build dependency information in the VDE database.

QUALIFIERS

/ADD (default)
/REMOVE

Controls whether the modules specified in the parameter list are added to or removed from the build-step definition. The /ADD qualifier adds the specified modules to the build step. The /REMOVE qualifier removes the specified modules from the build step. To set the default for the command as a whole, specify the qualifier after the MODIFY STEP keywords. To override that default for a single module, specify the qualifier after the **mod-name** parameter you choose to add to or remove from the build step. If you do not specify one of these qualifiers, modules are added to the build step.

Of /ADD and /REMOVE, /ADD is the default.

/COMPILE (default)
/COPY
/LINK

Specifies whether the build step to be modified is a compile step, copy step, or link step. /COMPILE modifies a compile step, /COPY modifies a copy step, and /LINK modifies a link step for the specified module.

Of /COMPILE, /COPY and /LINK, /COMPILE is the default.

/CONFIRM (default)
/NOCONFIRM

Controls whether VDE asks you to confirm that you want the build-step definition modified. The /CONFIRM qualifier causes VDE to print a message asking whether you want the build step modified. If you answer YES (or Y), the build-step definition is modified. If you answer NO (or N), the build-step definition is not modified. The /CONFIRM qualifier also causes VDE to ask you to confirm the addition or removal of each individual input or output module for the build step. The /NOCONFIRM qualifier causes VDE to modify the build-step definition without asking for confirmation.

/INPUT (default)
/OUTPUT

Specifies whether the **mod-name** parameters are inputs to or outputs from the build step. The /INPUT qualifier designates the specified modules as inputs in the build-step definition. The /OUTPUT qualifier

designates the specified modules as outputs in the build-step definition. To set the default for the command as a whole, specify the qualifier after the MODIFY STEP keywords. To override that default for a single module, specify the qualifier after the **mod-name** parameter you choose to mark as an input or output.

Of /INPUT and /OUTPUT, /INPUT is the default.

/LOG (default)
/NOLOG

Controls whether log messages are printed after the build step definition is modified in the database. The /LOG qualifier causes the messages to be printed and the /NOLOG qualifier suppresses them. The messages indicate that the build step has been modified, that the specified modules have been added to or removed from the step, and that the database transaction has committed successfully.

EXAMPLES

1

```
VDE> MODIFY STEP MOD4.FOR = FOO.FOR/INPUT, MOD4.ANA/OUTPUT
%VDE-I-COMSTEPMOD, compile step for module [CODE]MOD4.FOR modified in database
%VDE-I-MODINSTEP, module [CODE]FOO.FOR is an input to the build step
%VDE-I-MODOUTSTEP, module [CODE]MOD4.ANA is an output from the build step
%VDE-I-COMMIT, database transaction has successfully committed
VDE> SHOW STEP/FULL MOD4.FOR
Compile Step for module [CODE]MOD4.FOR
   Dependency links for stream V2.1 follow
   Inputs to the build step:
      Module [CODE]MOD4.FOR
      Module [CODE]FOO.FOR
   Outputs from the build step:
      Module [CODE]MOD4.$OBJ
      Module [CODE]MOD4.LIS
      Module [CODE]MOD4.ANA
   Build step has 2 inputs and 3 outputs

VDE>
```

This example modifies the build dependency information for the build step that compiles module MOD4.FOR. The command adds module FOO.FOR as an input to the step and adds module MOD4.ANA as an output from the step. The SHOW STEP command shows that the new input and output modules have been added to the build dependency information for the step.

```
2  VDE> MODIFY STEP/LINK MOD1.EXE = FOO.$OBJ/REMOVE/INPUT, MOD1.MAP/ADD/OUTPUT
   %VDE-I-LNKSTEPMOD, link step for module [CODE]MOD1.EXE modified in database
   %VDE-I-MODREMINSTEP, module [CODE]FOO.$OBJ removed as an input to build step
   %VDE-I-MODOUTSTEP, module [CODE]MOD1.MAP is an output from the build step
   %VDE-I-COMMIT, database transaction has successfully committed
   VDE> SHOW STEP/LINK/FULL MOD1.EXE
   Link Step for module [CODE]MOD1.EXE
      Dependency links for stream V2.1 follow
      Inputs to the build step:
         Module [CODE]MOD1.$OBJ
         Module [CODE]MOD2.$OBJ
      Outputs from the build step:
         Module [CODE]MOD1.EXE
         Module [CODE]MOD1.MAP
      Build step has 2 inputs and 2 outputs

   VDE>
```

This MODIFY STEP command modifies the build dependency information for the build step that links module MOD1.EXE. The command removes FOO.$OBJ as an input to the build step and adds MOD1.MAP as an output from the step. The SHOW STEP command confirms that the changes have been made to the step's build dependency information.

MODIFY STREAM

Modifies the attributes of one or more development streams in the VDE database.

Requires CRESTRM or MODSTRM privilege.

FORMAT

MODIFY STREAM *stream-name [, stream-name...]*

PARAMETER

stream-name

The name of a stream in the current VDE library whose attributes are to be modified. You can modify more than one stream at a time by using wildcard characters in the stream name. The percent sign (%) in a stream name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. VDE modifies the attributes of those streams whose names match the wildcard pattern.

The stream name can be up to 39 characters long. It must follow VDE name syntax. Unlike other VDE names, the stream name can contain periods (.). The use of the facility prefix *VDE$* is expressly prohibited.

QUALIFIERS

/CMS_CLASS
/NOCMS_CLASS

Sets or clears the CMS-class attribute of the stream. The /CMS_CLASS qualifier causes VDE to create a CMS class with the same name as the stream in each facility's CMS library and to then maintain that class when the REPLACE command and certain other commands change what module generations belong to the stream. The /NOCMS_CLASS qualifier causes VDE to not maintain a CMS class for the stream.

The /CMS_CLASS qualifier interacts with the /INIT_CMS_CLASS and /NOINIT_CMS_CLASS qualifiers; see the descriptions of these qualifiers for details.

You must have the CRESTRM privilege to use these qualifiers.

/CONFERENCE=file-spec
/NOCONFERENCE

Specifies whether a Notes conference for replacement information should be associated with the new stream. The /CONFERENCE qualifier specifies that the notes conference given by the **file-spec** parameter should be used to record all replacements into the stream. VDE then enters a new note into the conference for each queued or immediate replacement and a reply to that note when a queued replacement is actually performed. The /NOCONFERENCE qualifier specifies that no such notes conference should be used.

Under normal circumstances, just the file name should be specified in the **file-spec** parameter. VDE applies defaults to the notes conference file specification based on the translation of the logical name VDE$NOTES_ LIBRARY. If this logical name is not defined, the default file specification ''NOTES$LIBRARY:.NOTE'' is applied to the **file-spec** parameter.

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each stream definition modified. The /CONFIRM qualifier causes VDE to print a message for the specified stream giving the name of the stream and asking whether you want to modify that stream. If you answer YES (or Y), its attributes are modified. If you answer NO (or N), its attributes are not modified. The /NOCONFIRM qualifier causes VDE to modify the attributes of the stream without asking for confirmation.

/DELETE
/NODELETE (default)

Sets or clears the **delete** attribute for the stream. This attribute allows the stream to be deleted with the DELETE STREAM command. The /DELETE qualifier sets the delete attribute so that the stream can be deleted. The /NODELETE qualifier clears the delete attribute so that the stream cannot be deleted without first resetting the attribute. You must have the CRESTRM privilege to use these qualifiers.

/LOG (default)
/NOLOG

Controls whether log messages are printed after the stream definitions are modified. The /LOG qualifier causes such messages to be printed and /NOLOG suppresses them. The messages indicate that the stream definitions have been modified and that the database transaction has successfully committed.

/INIT_CMS_CLASS
/NOINIT_CMS_CLASS

Specifies whether VDE initializes the CMS class for the stream in each facility's CMS library when the /CMS_CLASS qualifier specifies that such a class should be maintained. /INIT_CMS_CLASS sets an attribute that causes the MODIFY STREAM command with the /CMS_CLASS qualifier to create and populate a CMS class with the same name as the stream in each facility. /NOINIT_CMS_CLASS clears this attribute so that MODIFY STREAM does not create or populate such CMS classes. /NOINIT_CMS_ CLASS is useful for very large VDE libraries where the initialization of CMS classes take an excessively long time. For such libraries, it is better to use a separate command procedure to initialize the desired CMS classes; such a procedure can process the facilities of your library in parallel for faster completion.

You must have the CRESTRM privilege to use these qualifiers.

/NAME=stream-name

Specifies a new name for the stream. The stream name can be up to 39 characters long and must follow normal VDE name syntax, except that a stream name can contain periods (.) . After the stream name has been

changed, you must refer to the stream by its new name in all subsequent VDE commands. The old name is lost from the database.

/NOSUCCESSOR[=(succ-name [, succ-name...])]

Specifies streams that should no longer be successors to the stream being modified. A successor to a stream is another stream to which source changes are propagated by the REPLACE command. Each **succ-name** parameter specifies the name of a stream that is currently a successor to the stream given by the **stream-name** parameter, but that no longer should be a successor. The specified streams are thus removed from the list of successors stored in the database.

If the list of **succ-name** parameters is omitted, all current successors of the stream being modified are removed from that stream's successor list.

You must have the CRESTRM privilege to use this qualifier.

/NOTIFY=(username [,username...])
/NONOTIFY[=(username [,username...])]

The /NOTIFY qualifier specifies a list of users to notify of all replacements into the specified stream, and of all facility creation operations. VDE appends these users to the stream's existing notification list. Each **username** parameter gives the OpenVMS username or mail address of a user who should be notified by mail message each time a module is replaced into the stream.

Each **username** parameter can specify the username of a user already recorded in the VDE database, a OpenVMS mail address with one or more node names (such as NODE::SMITH), a mail forwarding alias, or a defined logical name bound to a list of users to notify. If you specify a logical name, keep in mind that the logical name must be defined when it is later used to send notification messages. It should thus be a group or system logical name or a logical name that is in some other way defined for all users of the VDE library.

The /NONOTIFY qualifier specifies that a list of users already on the stream's notification list be removed from that notification list. Each **username** parameter gives the OpenVMS username or mail address of a user to remove from the notification list. If you omit the **username** parameters, VDE removes the entire notification list for the stream.

If you specify both qualifiers, VDE removes the users specified with the /NONOTIFY qualifier from the notification list before adding the users specified with the /NOTIFY qualifier.

If you specify only one **username** parameter with either qualifier, you may omit the surrounding parentheses.

/OPEN
/FROZEN
/CLOSED

Sets the status of the stream to open, frozen, or permanently closed. If you specify the /OPEN qualifier, all users can reserve and replace source modules from the stream and thereby modify the contents of the stream.

If you specify the /FROZEN qualifier, all users may reserve modules from the stream. However, a user must have the FROZEN privilege to replace modules into the stream; non-privileged users cannot change the contents of the stream. You can later change the status to open with another MODIFY STREAM command, so the stream is not permanently frozen.

If you specify the /CLOSED qualifier, modules cannot be reserved from or replaced into the stream. This status cannot be changed with a subsequent MODIFY STREAM command unless you have the UNCLOSE privilege. Use the /CLOSED qualifier when you never want the contents of the stream to change again, for example, when capturing a base-level of your software system or when capturing the final state of a released version of your software.

You must have the UNCLOSE privilege to use the /OPEN or /FROZEN qualifier for a closed stream.

/OWNER=username
/NOOWNER

The /OWNER qualifier specifies the owner of the specified stream. The **username** parameter gives the OpenVMS username of a user who should be recorded as the ''owner'' of the stream. VDE assigns no special meaning to the owner of a stream, but the SHOW STREAM command displays the owner. The owner could thus be the person to contact with questions about the stream.

The /NOOWNER qualifier specifies that the stream has no owner. Both qualifiers remove the previous owner's ownership of the stream.

/POPULATE
/NOPOPULATE

Specifies whether the stream is allowed to be populated with generations from another stream using the POPULATE STREAM command. /POPULATE allows subsequent use of the POPULATE STREAM command to populate the stream and /NOPOPULATE disallows such use. You must have the CRESTRM privilege to use these qualifiers.

/PROCESS_COUNT=proc-count

Specifies the default number of processes to execute build jobs for the stream. This parameter determines the degree of parallelism for the stream's build jobs. See the description of the BUILD command for a detailed description of this qualifier.

/QUEUE=(queue-name [,queue-name...])

Specifies the default batch queues to which build jobs for the stream are submitted. See the description of the BUILD command for a detailed description of this qualifier and for a description of its interaction with the /PROCESS_COUNT qualifier.

/REMARK="string"

Specifies a remark string for the stream. The quoted remark string, which can be up to 132 characters long, may contain any explanatory remark about the stream that you want stored in the VDE database. Typically the remark describes what version or base-level of your system the stream represents. This string replaces the remark string from the original CREATE STREAM command or from any previous MODIFY STREAM command.

/REPLACEMENT=keyword

Sets or clears the **queued-replacement** attribute for the stream. This attribute controls whether the REPLACE command permits immediate replacements into the stream or requires queued replacements. If the **keyword** parameter is IMMEDIATE, the REPLACE command will immediately replace modules into the stream by default. If the parameter is QUEUE, the REPLACE command will always queue replacements for the stream so that modules are not replaced into the stream until a privileged user enters a PERFORM REPLACEMENT command.

/REVIEWER=(username [,username...])
/NOREVIEWER[=(username [,username...])]

The /REVIEWER qualifier specifies a list of reviewers for all module replacements into the specified stream. VDE appends these reviewers to the stream's existing reviewer list. Each **username** parameter gives the OpenVMS username of a user who should review all replacements into the stream. The user must already be recorded in the VDE database. When a module is replaced into the stream, VDE notifies the user of the replacement by mail message. If the replacement is queued, the user should then review the queued replacement before it is performed. If the replacement is immediate, VDE notifies the user of the new module generation that was created, and the user may then inspect that generation.

The /NOREVIEWER qualifier specifies that a list of existing reviewers for the specified stream be removed from the reviewer list. Each **username** parameter gives the OpenVMS username of a user who should be removed from the reviewer list. If you omit the **username** parameters, VDE removes the entire reviewer list for the stream.

If you specify both qualifiers, VDE removes the users specified with the /NOREVIEWER qualifier from the reviewer list before adding the users specified with the /REVIEWER qualifier.

If you specify only one **username** parameter with either qualifier, you may omit the surrounding parentheses.

/STEP_DELAY=time-interval

Specifies the default step-delay time interval used by build jobs for the stream. See the description of the BUILD command for a detailed description of this qualifier.

/SUCCESSOR=(succ-name [, succ-name...])

Specifies new successors to the stream being modified. A successor to a stream is another stream to which source changes are propagated by the REPLACE command. Each **succ-name** parameter specifies the name of a stream that becomes a successor to the stream given by the **stream-name** parameter. The specified successors are added to the list of successors already stored in the database. You must have the CRESTRM privilege to use this qualifier.

/TIMEOUT=time-interval

Specifies the default timeout interval used by build jobs for the stream. See the description of the BUILD command for a detailed description of this qualifier.

EXAMPLES

1

```
VDE> MODIFY STREAM V3.3-FT1 /REMARK="New remark string"
%VDE-I-STRMMOD, information for stream V3.3-FT1 modified in the database
%VDE-I-COMMIT, database transaction has successfully committed
```

In this example, the user specifies a new remark string for stream V3.3-FT1. This remark string can be retrieved with a subsequent SHOW STREAM command.

2

```
VDE> MODIFY STREAM/NAME=V5.3 HICKORY
%VDE-I-STRMMOD, information for stream V5.3 modified in the database
%VDE-I-COMMIT, database transaction has successfully committed
```

In this example, the name of stream HICKORY is changed to V5.3. You must refer to the stream by this new name in all subsequent VDE commands.

3

```
VDE> MODIFY STREAM/CONFIRM BLVL-7/REMARK="New remark string", -
_VDE>                      BLVL-9/REMARK="New remark string"
Modify stream BLVL-7 ? [No]: Y
%VDE-I-STRMMOD, stream BLVL-7 modified in the database
Modify stream BLVL-9 ? [No]: N
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example illustrates the /CONFIRM qualifier. This qualifier causes VDE to ask the user whether each specified stream should be modified. For stream BLVL-7, the user answers Y (for YES) and the remark string for that stream is modified in the database, as the log message shows. For stream BLVL-9, the user answers N (for NO) and that stream is not modified. Hence there is no log message for that stream. The final log message indicates that the database transaction for the whole command has successfully committed.

MODIFY USER

Modifies one or more attributes of the database record of one or more users.

Requires CREUSR privilege.

FORMAT

MODIFY USER *username [, username...]*

PARAMETER

username

The OpenVMS username of the user whose record is to be modified in the current VDE library. You can modify more than one user record by using wildcard characters in the username. The percent sign (%) in a username matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. The attributes of those user records whose usernames match the wildcard pattern are modified.

QUALIFIERS

/AUTH_PRIVILEGES=(priv [,priv...])

Specifies a new set of authorized privileges that the user is allowed to set with the SET PRIVILEGE command. The privilege list you create replaces the user's current privilege list. Therefore, you must specify all authorized privileges you want the user to have.

Each parameter to this qualifier names a single privilege. If only one privilege is specified, the parentheses may be omitted.

For a list of privileges, see Table 3–5 in SET PRIVILEGES.

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each user record modified. The /CONFIRM qualifier causes VDE to print a message for each user giving the username and asking whether you want to modify the user record. If you answer YES (or Y), the user record is modified. If you answer NO (or N), the user record is not modified. The /NOCONFIRM qualifier causes VDE to modify the user record without asking for confirmation.

/DEF_PRIVILEGES=(priv [,priv...])

Specifies the default privileges which are enabled each time the user starts a new VDE session. Each default privilege specified also becomes an authorized privilege for the user. The privilege list you create replaces the user's current privilege list. Therefore, you must specify all default privileges you want the user to have.

Each parameter to this qualifier names a single privilege. If only one privilege is specified, the parentheses may be omitted. For a list of the privilege names you can specify, see the SET PRIVILEGES command.

/LOG (default)
/NOLOG

Controls whether log messages are printed after the user records are modified. The /LOG qualifier causes such messages to be printed and /NOLOG suppresses them. The messages indicate that the user records have been modified and that the database transaction has successfully committed.

/NAME="string"

Specifies a full name for the user. The quoted name string, which can be up to 40 characters long, should contain a new full name string (first name, middle initial, and last name) for the user.

/NOTIFY=(username [,username...])
/NONOTIFY[=(username [,username...])]

The /NOTIFY qualifier specifies a list of users to notify each time the specified user replaces a module into the VDE library. VDE appends these users to the specified user's existing notification list. Each **username** parameter gives the OpenVMS username or mail address of a user who should be notified by mail message each time the specified user replaces a module into the library.

Each **username** parameter can specify the username of a user already recorded in the VDE database, a OpenVMS mail address with one or more node names (such as NODE::SMITH), a mail forwarding alias, or a defined logical name bound to a list of users to notify. If you specify a logical name, keep in mind that the logical name must be defined when it is later used to send notification messages. It should thus be a group or system logical name or a logical name that is in some other way defined for all users of the VDE library.

The /NONOTIFY qualifier specifies that a list of users already on the specified user's notification list be removed from that notification list. Each **username** parameter gives the OpenVMS username or mail address of a user to remove from the notification list. If you omit the **username** parameters, VDE removes the entire notification list.

If you specify both qualifiers, VDE removes the users specified with the /NONOTIFY qualifier from the notification list before adding the users specified with the /NOTIFY qualifier.

If you specify only one **username** parameter with either qualifier, you may omit the surrounding parentheses.

/REMARK="string"

Specifies a remark string to be stored for the user. The quoted remark string, which can be up to 132 characters long, may contain any explanatory remark about the user that you want stored in the VDE database. This string replaces the remark string from the original CREATE USER command or from any previous MODIFY USER command. Typically the remark describes the user's affiliation or function within the organization.

/REVIEWER=(username [,username...])
/NOREVIEWER[=(username [,username...])]

The /REVIEWER qualifier specifies a list of reviewers for all changes replaced into the VDE library by the specified user. VDE appends these

reviewers to the user's existing reviewer list. Each **username** parameter gives the OpenVMS username of a user who should review all changes created by the specified user. The new reviewer must already be recorded in the VDE database. When the specified user replaces a module into the VDE library, VDE notifies the reviewer of the change by mail message. If the replacement is queued, the reviewer should then review the queued replacement before it is performed. If the replacement is immediate, VDE notifies the reviewer of the new module generation that was created, and he or she may then inspect that generation. This qualifier can be used to set up ''buddy systems'' where two users always review each other's code.

The /NOREVIEWER qualifier specifies that a list of existing reviewers for the specified user be removed from the reviewer list. Each **username** parameter gives the OpenVMS username of a user who should be removed from the reviewer list. If you omit the **username** parameters, VDE removes the entire reviewer list for the user.

If you specify both qualifiers, VDE removes the users specified with the /NOREVIEWER qualifier from the reviewer list before adding the users specified with the /REVIEWER qualifier.

If you specify only one **username** parameter with either qualifier, you may omit the surrounding parentheses.

/USERNAME=username

Specifies a new OpenVMS username for the user. The new username can be up to 12 characters long and must follow normal VDE name syntax. If you change a username in the VDE database you should also remember to have your system manager change it in the system User Authorization File. After the username is changed, you must refer to the user by the new name in all subsequent VDE commands and the user must access the database with the new username.

EXAMPLES

1

```
VDE> MODIFY USER SMITH /USERNAME=BROWN /NAME="Jane S. Brown"
%VDE-I-USERMOD, information for user BROWN modified in the database
%VDE-I-COMMIT, database transaction has successfully committed
```

In this example, Jane Smith has changed her name. The MODIFY USER command is used to enter her new OpenVMS username and full name string into the VDE database. Thereafter, Jane must access the VDE library as user BROWN and she must be referred to as BROWN on subsequent VDE commands. The log messages confirm that her user record was modified and that the database transaction successfully completed.

2

```
VDE> MODIFY USER DOE /REMARK="OpenVMS Documentation Supervisor" -
_VDE>       /DEF_PRIVILEGES=BUILD /NOLOG
VDE>
```

Here John Doe has been promoted to Documentation Supervisor. The MODIFY USER command changes his remark string to indicate his new role and reduces his default privileges to only the BUILD privilege. In this example, the /NOLOG qualifier suppresses the log messages.

PERFORM REPLACEMENT

Performs one or more queued replacements so that the corresponding modules are replaced into the VDE library.

Requires PERFREP privilege.

FORMAT

PERFORM REPLACEMENT *rep-name [, rep-name...]*

PARAMETER

rep-name

The name of a queued replacement to be performed. You can perform more than one replacement at a time by using wildcard characters in the replacement name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Those queued replacements whose names match the wildcard pattern are performed.

DESCRIPTION

The PERFORM REPLACEMENT command replaces the modules that are members of a specified queued replacement into the VDE library. Once this command has completed, each member module has a new generation in the library and is no longer marked as reserved in the affected streams. In addition, the replacement is no longer queued in the VDE database.

The PERFORM REPLACEMENT command only performs those replacements that match the **rep-name** parameters and that belong to your default stream or to another stream specified with the /STREAM qualifier. To perform replacements belonging to more than one stream, you can use wildcard characters in the stream name you specify with the /STREAM qualifier.

The PERFORM REPLACEMENT command is normally used by a user who is authorized to determine when changed modules should be replaced into the VDE library. Other developers reserve source modules, make changes to them, and queue them for replacement with the REPLACE command. The REPLACE command queues modules for replacement (as opposed to replacing them immediately), if the user specifies the /QUEUE qualifier or if any affected development stream has the queued-replacement attribute. Once a replacement is queued, it can optionally be reviewed using the REVIEW REPLACEMENT command and its subcommands. When the authorized user decides that the modules that are members of the queued replacement are ready to be replaced into the VDE library, the user enters the PERFORM REPLACEMENT command to actually perform the replacement. When replacements are queued, it is the PERFORM REPLACEMENT command that actually creates new generations for the modules and that terminates the module reservations.

Whether you should use queued replacements or not is a policy decision that may be different for different VDE libraries or for different development stream within a given library. If you decide to always use immediate replacements, the REPLACE command will perform the actual replacement and you do not use the PERFORM REPLACEMENT command.

QUALIFIERS

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each replacement performed. The /CONFIRM qualifier causes VDE to print a message for each replacement asking whether you want that replacement performed. If you answer YES (or Y), that replacement is performed. If you answer NO (or N), the replacement is not performed. If you enter a replacement name with a wildcard specification, VDE asks you to confirm that VDE should perform each replacement whose name matches the wildcard specification. The /NOCONFIRM qualifier causes VDE to perform the specified replacements without asking for confirmation.

/LOG (default)
/NOLOG

Controls whether log messages are printed after each replacement has been performed. The /LOG qualifier causes such messages to be printed and the /NOLOG qualifier suppresses them. These messages indicate that the associated modules have been replaced, that the replacement has been performed, and that the database transaction has successfully committed.

/STREAM=stream-name

Specifies that only those replacements that belong to certain streams be performed. The **stream-name** parameter specifies a stream name and can contain wildcard characters. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. VDE performs a given replacement only if it belongs to a stream whose name matches the **stream-name** parameter. If you omit this qualifier, VDE performs only those replacements that belong to your default stream.

For each replacement that matches a **rep-name** parameter but does not belong to a stream matching the **stream-name** parameter, VDE prints an informational message saying that the replacement is not performed.

EXAMPLES

1
```
VDE> PERFORM REPLACEMENT SMITH-2
%VDE-I-REPLACED, generation [FACNAM]A.REQ;9(9) replaced into stream V2.0
%VDE-I-REPLACED, generation [FACNAM]B.REQ;6(6) replaced into stream V2.0
%VDE-I-REPPERF, replacement SMITH-2 performed
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example performs the queued replacement SMITH–2. The PERFORM REPLACEMENT command replaces the two modules that are members of the replacement, A.REQ and B.REQ in facility FACNAM, into

the VDE library. Thereafter, these two modules are no longer reserved. The command also deletes replacement SMITH–2 from the VDE database.

POPULATE STREAM

Populates a development stream with generations from another stream.

Requires PERFREP privilege.

FORMAT

POPULATE STREAM *stream-name [, stream-name...]*

PARAMETER

stream-name

The name of a target stream to be populated with the module generations from another stream. You can populate more than one stream at a time by using wildcard characters in the stream name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Those streams whose names match the wildcard pattern are populated.

DESCRIPTION

The POPULATE STREAM command populates a specified development stream (the target stream) with the latest generations from another stream (the source stream). The target stream is specified with a command parameter and the source stream is specified with the /FROM_ STREAM qualifier. The latest generation of each module in the source stream is inserted into the target stream and any modules not represented in the source stream are removed from the target stream. After the command completes, the target stream contains exactly the same generations as the source stream.

Using the CREATE STREAM command to create a new stream with the same contents as a parent stream can be a time-consuming operation if the VDE library is large. Using the POPULATE STREAM command to update an existing stream whose contents are already close to the desired source stream is generally a much faster way to make a copy of a stream. This command allows you to create new base-level streams by reusing old base-level streams that are no longer needed, thus saving both time and database space.

The POPULATE STREAM command updates only the VDE database. It does not insert the updated generations into the CMS class for the stream, not even if the attribute to maintain the CMS class is set for the target stream. Neither does the command update the source directories for the stream. If such operations are needed, they must be done manually outside of VDE. Alternatively you can use the INSERT GENERATION command to insert generations into a stream while maintaining CMS classes and stream source directories. Because it does more work, the INSERT GENERATION command is generally much slower than the POPULATE STREAM command.

POPULATE STREAM

VDE populates the target streams in three phases. First, it determines what streams to populate using a read-only database transaction. Second, it determines the differences between the source and target streams for one target stream. This step is done using a read-only transaction. Third, VDE loops over the differences it found to update the generation membership of the target stream. This phase gives the target stream the same generation membership as the source stream had during the preceding read-only transaction. During this phase, VDE updates 20 modules at a time (by default), commits the transaction, and repeats the process until all modules are updated. VDE uses multiple short read-write transactions in order not to lock the database against other users more than a few seconds at a time. After the target stream is fully populated, VDE loops to populate the next stream until none remain.

You can only populate an open stream if the **populate** attribute is set for the stream. If this attribute is not set, you must set it with the MODIFY STREAM /POPULATE command before you can populate the stream. You can only populate a frozen stream if the **populate** attribute is set and you have the FROZEN privilege. You cannot populate a closed stream.

QUALIFIERS

/COMMIT_LIMIT=number

Specifies the maximum number of modules VDE will update in, insert into, or remove from the target stream before committing the database transaction and starting a new transaction. The **number** parameter must be an integer value in the range from 1 to 1,000,000.

Specification of a smaller value reduces the interval during which the database is locked, allowing access to the database by other users. Specification of a larger value does reduce the overhead involved with recycling locks and repeated commits, but restricts the access to the database by other users.

By default, VDE commits and restarts a read-write transaction for every 20 modules.

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each target stream populated. The /CONFIRM qualifier causes VDE to print a message for each stream asking whether you want that stream populated. If you answer YES (or Y), that stream is populated from the source stream. If you answer NO (or N), the stream is not populated. If you enter a stream name with a wildcard specification, VDE asks you to confirm the populate action for each stream whose name matches the wildcard specification. The /NOCONFIRM qualifier causes VDE to populate the specified streams without asking for confirmation.

/FROM_STREAM=stream-name

Specifies the source stream from which the target stream is to be populated. VDE makes the generation membership of the target stream identical to that of this source stream. This qualifier is required; if you omit it, VDE prompts you for the name of the source stream.

/LOG (default)
/NOLOG

Controls whether log messages are printed after each stream has been populated. The /LOG qualifier causes such messages to be printed and the /NOLOG qualifier suppresses them. These messages indicate the number of modules to update, insert, or remove to populate the target stream, that the stream has been populated from the source stream, and that the database transaction has successfully committed.

EXAMPLES

1

```
VDE> POPULATE STREAM BLD_12/FROM_STREAM=BIRCH
%VDE-I-STRMPOPCOUNT, populate module counts: update 5, insert 2, remove 1
%VDE-I-STRMPOP, stream BLD_12 populated from stream BIRCH
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

In this example, stream BLD_12 is populated from stream BIRCH. Five module generations in stream BLD_12 are updated, two are inserted, and one is removed to make the generation membership of that stream identical to stream BIRCH. The log messages confirm that the stream was populated and that the database transaction successfully completed.

REJECT REPLACEMENT

Marks a queued replacement as rejected in the VDE database. You can only mark a replacement as rejected if you are a reviewer for that replacement.

Requires RESREP privilege.

FORMAT

REJECT REPLACEMENT *rep-name [, rep-name...] ["remark"]*

PARAMETER

rep-name

The name of a queued replacement to be marked as rejected. You can reject more than one replacement at a time by using wildcard characters in the replacement name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Those queued replacements whose names match the wildcard pattern are marked as rejected.

"remark"

Specifies a one-line remark for the replacement. Typically this remark summarizes why you reject the replacement. Specifying the **remark** parameter is equivalent to specifying the /REMARK qualifier after the REJECT REPLACEMENT keywords. This parameter or the /REMARK qualifier is required; if you do not specify either, VDE prompts you for a remark string.

DESCRIPTION

The REJECT REPLACEMENT command updates the VDE database to record that you have reviewed the code changes in a specified queued replacement and that you judge those changes not acceptable for replacement into the VDE library. You must be a reviewer for a replacement in order to mark it as rejected.

The REJECT REPLACEMENT command allows you to attach a comment file to the queued replacement. The comment file typically explains why you reject the replacement. If you have already created such a file, use the /INPUT qualifier to copy it to the replacement's staging area. If you omit the /INPUT qualifier, VDE asks you whether you want to edit a comment file for the replacement. If you respond YES (the default answer), VDE invokes your preferred editor to create the comment file. If you specify both the /INPUT and the /EDIT qualifier, VDE copies the input file to the replacement's staging area and then invokes your editor to further modify that file. If you have already created a comment file for the replacement with an earlier ACCEPT or REJECT command, VDE lets you edit that file again. This capability is useful if you change your mind or want to add more information.

If you wish to mark a replacement as acceptable, use the ACCEPT REPLACEMENT command. If you wish to see the review status of a queued replacement, use the SHOW REPLACEMENT command.

QUALIFIERS

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each replacement rejected. The /CONFIRM qualifier causes VDE to print a message for each replacement asking whether you want to reject that replacement. If you answer YES (or Y), that replacement is marked as rejected. If you answer NO (or N), the replacement status is not changed. If you enter a replacement name with a wildcard specification, VDE asks you to confirm the rejection of each replacement whose name matches the wildcard specification. The /NOCONFIRM qualifier causes VDE to mark the specified replacements as rejected without asking for confirmation.

/EDIT

Specifies that VDE should invoke your preferred editor so that you can edit your comment file for each queued replacement. If such a comment file already exists (due to a previous ACCEPT or REJECT REPLACEMENT command for the same replacement) or if you specify an input file with the /INPUT qualifier, the editor modifies that file. Otherwise, the editor creates a new file in the replacement's staging directory. Its file name is your OpenVMS username and its file extension is VDE$COMMENT. The comment file typically contains comments for the author of the replacement and the other reviewers.

This qualifier is ignored if you are running in batch mode.

/INPUT=file-spec

Specifies a comment file for the queued replacement. VDE reads the text file given by the **file-spec** parameter and writes its contents into a new file in the replacement's staging directory. The file name of the new file is your OpenVMS username and its file extension is VDE$COMMENT. The comment file typically contains remarks to the author of the replacement or information for the other reviewers.

/LOG (default)
/NOLOG

Controls whether log messages are printed after each replacement has been marked as rejected. The /LOG qualifier causes such messages to be printed and the /NOLOG qualifier suppresses them. These messages indicate that the replacement has been rejected and that the database transaction has successfully committed.

/REMARK="string"

Specifies a one-line remark for the replacement. Typically this remark summarizes why you reject the replacement. For longer comments, you may create a comment file with the /INPUT or /EDIT qualifier. The /REMARK qualifier (or the remark parameter at the end of the command) is required; if you do not specify it, VDE prompts you for a remark string.

/USERNAME=username

Specifies that the replacement be marked as rejected on behalf of another user. The **username** parameter specifies the OpenVMS username of the other user. That user must be a reviewer for the specified replacement. You must have the USERNAME privilege to use this qualifier.

EXAMPLES

1

```
VDE> REJECT REPLACEMENT BROWN-3 "Bug found in sort code"
%VDE-I-REPREJECT, replacement BROWN-3 marked as rejected in database
%VDE-I-COMMIT, database transaction has successfully committed
Do you wish to edit a comment file for replacement BROWN-3 ? [Yes]: YES

   ...Comment file edited here...

VDE>
```

This command marks that the current user rejects queued replacement BROWN–3. The remark string indicates the reason for the rejection. Since the /INPUT qualifier is omitted, VDE asks the user whether he or she wants to edit a comment file. The user answers YES and is placed in an editing session.

2

```
VDE> REJECT REPLACEMENT SMITH-1/INPUT=COMMENT_FILE.TXT
Please supply required information for replacement SMITH-1:
Enter a remark for the rejection: Code does not conform to coding conventions
%VDE-I-REPREJECT, replacement SMITH-1 marked as rejected in database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

In this example, the user marks replacement SMITH–1 as rejected and provides a comment file that gives additional information to the user who queued the replacement. The comment file may explain, for example, what defects in the code caused it to be rejected. Since the /REMARK qualifier is omitted, VDE prompts for a remark string.

REMOVE GENERATION

Removes a module from a specified development stream so that the module no longer has a latest generation in the stream.

Requires PERFREP privilege.

FORMAT

REMOVE GENERATION *mod-name [, mod-name...]*

PARAMETER

mod-name

The name of a source module whose latest generation is to be removed from the specified development stream. The module name consists of an optional facility name enclosed in square brackets, a module name, and an optional type name preceded by a period, such as [FACIL]MOD1.MAR. If no facility name is specified, the default facility is assumed. If no type name is specified, the latest generations of all source modules of the specified module name in the given facility are removed.

You can remove more than one module from the stream by using wildcard characters in any of the three components of the module name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. VDE removes each module whose name matches the wildcard pattern from the stream.

DESCRIPTION

The REMOVE GENERATION command removes a specified module from a specified development stream so that the module no longer has a latest generation in that stream. The result is that the module no longer ''belongs'' to that stream and can no longer be fetched, reserved, or built in the stream.

The removed generation remains in the VDE database and the VDE library, and it can still be the latest generation of the module for some other stream. The REMOVE GENERATION command thus breaks the association between the module generation and the specified stream but does not delete the generation.

If the specified module is currently reserved in the specified stream, VDE terminates that reservation since it is based on a module generation that no longer is the latest generation for the stream. VDE sends a mail message to notify the owner of the reservation that the reservation is lost.

QUALIFIERS

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each generation removed. The /CONFIRM qualifier causes VDE to print a message for the latest generation of each module giving the name of the generation and

asking whether you want to remove the generation. If you answer YES (or Y), the generation is removed. If you answer NO (or N), the generation is not removed. The /NOCONFIRM qualifier causes VDE to remove each generation without asking for confirmation.

/LOG (default)
/NOLOG

Controls whether log messages are printed after generations are removed from streams. The /LOG qualifier causes such messages to be printed and /NOLOG suppresses them. The messages indicate that the generations have been removed from the specified development stream and that the database transaction has successfully committed.

/REMARK="string"

Specifies a remark string that explains why you removed the generation from the specified stream. The quoted remark string can be up to 132 characters long.

/STREAM=stream-name

Specifies that the generation be removed from the development stream given by the **stream-name** parameter. If this qualifier is omitted, VDE prompts you for the stream name.

EXAMPLES

1

```
VDE> REMOVE GENERATION [FACIL]MODNAM.C /STREAM=MANGO
%VDE-I-GENREM, generation [FACIL]MODNAM.C;4(4) removed from stream MANGO
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example removes the latest generation of module [FACIL]MODNAM.C from stream MANGO. The log message shows that generation 4 had been the latest generation of the module in that stream, and is now removed. After the command has completed, no generation of the module belongs to stream MANGO.

REPLACE

Returns each specified module to the library and creates a new generation of the module. Each module must be a source module previously reserved with the RESERVE command.

Requires RESREP privilege.

FORMAT

REPLACE *mod-name [, mod-name...] ["remark"]*

PARAMETER

mod-name

Specifies a reserved source module to be replaced into the library. The module name consists of an optional facility name enclosed in square brackets, a module name, and an optional type name preceded by a period (such as [FACIL]MOD1.MAR). If the facility name is omitted, the module is assumed to belong to the default facility. If the type name is omitted, all source modules with the specified module name in the given facility are replaced.

You can replace more than one module by using wildcard characters in any of the three components of the module name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. VDE replaces the source modules whose names match the wildcard pattern.

You can also replace multiple modules by specifying the name of a source group instead of a module name. Source groups are created with the CREATE GROUP command. If you specify a group name, each module that is a member of the group is replaced.

If you use the /SESSION qualifier, those reservations that are members of the specified reservation session are replaced. In this case, you must omit the **mod-name** parameters from the command.

"remark"

Specifies a remark string for the replacement. VDE stores this remark string with each module generation that the REPLACE command creates. Specifying the **remark** parameter is equivalent to specifying the /REMARK qualifier after the REPLACE keyword.

DESCRIPTION

For each module you have specified, the REPLACE command creates a new generation of the module by transferring a file from your default directory to the current VDE library, possibly by way of an intermediate staging area. This new generation becomes the most recent generation of the source module in the default development stream (or the stream specified with the /STREAM qualifier) and receives a new generation number. The new generation number is formed by adding 1 to the number of the previous generation. After creating the new module generation, the REPLACE command terminates the reservation of that module unless you specify the /RESERVE qualifier.

After the reserved module is replaced in the VDE library, VDE may delete the file used to create the module (as well as any earlier versions of the file in the same directory). The file is only deleted if the library administrator has defined a NOKEEP script for your VDE library and the specified development stream. If you specify either the /KEEP or the /RESERVE qualifier, VDE does not delete the file even if a NOKEEP script is defined.

The file in your default directory normally has the same name as the module in the library. However, you can direct VDE to replace a file with a different name or a file in another directory than your default directory by specifying the /INPUT qualifier.

VDE stores the following information about the file that creates the new generation: the creation date and time, the revision date and time, the record format, and other record attributes. When you fetch or reserve a generation of a module from a VDE library, the file that is placed in your directory has the same creation and revision times, record format, and record attributes as the file that was used to create the generation that you are fetching or reserving.

If you specify the /QUEUE qualifier or one of the streams that you are replacing the module into has the queued-replacement attribute, then the REPLACE command does not actually perform the replacement of your module. Instead it copies your files to a ‘‘staging area’’ on disk and queues the replacement in the VDE database. An authorized user must then enter a subsequent PERFORM REPLACEMENT command in order to actually perform the replacement. In this case, it is the PERFORM REPLACEMENT command the creates the new module generation and terminates the module reservation.

If you have more than one concurrent reservation of a module, you must specify the exact reservation to be replaced using the /IDENTIFICATION qualifier. Use the SHOW RESERVATION command to determine the identification of each reservation.

If you or another user has a concurrent reservation for the module being replaced and that reservation covers at least one stream affected by your replacement, then the REPLACE command terminates that reservation. VDE sends a mail message to notify the owner of that reservation that the reservation is lost.

To replace another user’s reservation, you must have the USERNAME privilege and use the /USERNAME qualifier to specify OpenVMS username of that user.

Source Tracking and Statistics Collection

VDE can collect replacement statistics and associated change-related information, including the reason for the change, the project the change is associated with, and can also be asked to add an entry to the source code tracking (SCT) DEC Notes conference describing the change.

Statistics collection is enabled or disabled on a library basis. If collection is enabled and the information is not provided via REPLACE qualifiers, VDE will prompt the user for the information during the replacement operation.

For further information, see the /FOLD, /OTHER, and /SCT qualifiers, as well as the /PROJECT qualifier.

Change Propagation to other Streams

Replacing a module makes the new module generation the latest generation in the default development stream or the stream specified with the /STREAM qualifier. That generation is thus the generation you get by default in a subsequent FETCH or RESERVE command, and is the generation that is compiled when you build your system in that stream.

If the specified stream has successors, the REPLACE command propagates the new module generation to all successor streams. For example, if stream V2 is a successor of stream V1, then any module that is replaced into stream V1 is also replaced in stream V2 automatically. If stream V2 has additional successors (such as stream V3), the new module generation is propagated to those streams, and so on recursively.

Propagation works as follows. If the latest generations of the module for the original stream and the successor stream are the same, then the new module generation created by the REPLACE command becomes the latest generation for both streams. Both streams are thus equally up-to-date with respect to the changes made to that module. However, if the latest generations of the module differ for the two streams, which means that the module has diverged between those two streams, VDE does not propagate the new module generation to the successor stream. Instead, it creates a ''fold'' record in the database to remind the developer that the change made to the module for the first stream must be manually ''folded'' into the successor stream. Outstanding folds can be displayed with the SHOW FOLD command and a fold record is deleted with the CANCEL FOLD command.

Replacing a Module with History or Notes Information

If you reserve a generation of a module with an embedded history and then replace it, the REPLACE command ignores the history; that is, VDE does not copy the history into your VDE library. If you add text to the file in or above the history (relative to #B), or in or below the history (relative to #H), the REPLACE command issues an error message and the command is not executed.

If you reserve a file with embedded notes and then replace it, the REPLACE command does not copy the notes to the VDE library. If, while editing the file, you insert text that looks like an embedded note, that text is deleted when the file is replaced.

QUALIFIERS

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each module replaced. The /CONFIRM qualifier causes VDE to print a message for each specified module asking whether you want to replace that module into the library. If you answer YES (or Y), it is replaced. If you answer NO (or N), it is not replaced. The /NOCONFIRM qualifier causes VDE to replace each specified module without asking for confirmation.

/DEFECTS_CORRECTED=error-count
/NODEFECTS_CORRECTED

Specifies the number of defects corrected by this replacement: these defects could due to problems with the specification, design, logic, implementation and/or testing. The **error-count** parameter is a value indicating the number of defects corrected.

/NODEFECTS_CORRECTED sets the number of defects corrected to zero.

This qualifier is part of the statistics collection component of VDE. Statistics collection is enabled or disabled on a library basis, using the MODIFY LIBRARY/STATISTICS command. If statistics collection is disabled, these qualifiers are ignored and have no adverse effects. This command is *required* when certain keywords are specified on the /REASONS qualifier; see Table 3–2 for further information.

If statistics collection is enabled on the current library and you do not specify the /REASONS qualifier, you will be prompted for the information.

/FOLD[=fold-ident]

Specifies that a fold record be cancelled when this replacement is performed. The **fold-ident** parameter gives the fold identifier of a fold record for the module being replaced and for the stream in which the replacement takes place. When VDE performs the replacement, it will cancel this fold record from the VDE database. You can omit the **fold-ident** parameter if there is only one fold record for the specified module and stream. You should use the /FOLD qualifier if the current replacement is done to satisfy an existing fold record in the database.

Also see /OTHER and /SCT.

/IDENTIFICATION=res-ident

Specifies the reservation to be replaced. This qualifier is required when you have multiple reservations of the same module in the default stream. The **res-ident** parameter is the reservation identifier of the reservation to be replaced. The reservation identifier is the identifier you specified or VDE assigned when you reserved the module, usually a small integer value. Use the SHOW RESERVATION command to determine the reservation identifier of each reservation.

/INFORMATION[=file-spec]
/NOINFORMATION

The /INFORMATION qualifier specifies that an information file be attached to the current replacement if it is a queued replacement. The **file-spec** parameter gives the OpenVMS file specification of the information file. If this parameter is omitted, VDE asks you whether you want to edit an information file. If you answer YES, VDE invokes your preferred editor to edit the information file. You may use the information file to pass information about the replacement to reviewers and others. The /NOINFORMATION qualifier specifies that you do not want to provide an information file.

If you omit both qualifiers, VDE's behavior depends on the library attribute that is set with the /ASK_INFO_FILE or cleared with the /NOASK_INFO_FILE qualifier to the CREATE or MODIFY LIBRARY command. If this attribute is set, VDE asks you whether you want to edit an information file. If this attribute is not set, VDE queues the replacement without an information file.

For immediate replacements, VDE ignores both qualifiers and does not create an information file.

To edit an information file through VDE, you should first declare your preferred editor with the SET EDITOR command. If you do not declare an editor, VDE invokes the system default text editor.

/INPUT=file-spec

Specifies a file to be used as input for the replacement operation. If you omit the directory specification from this qualifier, the file is taken from your default directory. If you omit the file name or file extension, the module name and module type specified in the command parameter are used by default.

/KEEP
/NOKEEP (default)

If the administrator for your VDE library has established a NOKEEP script that deletes files from your directory, this qualifier controls whether the file used to create the new module generation is deleted from your directory. The /KEEP qualifier keeps the file in your directory without deleting it. The /NOKEEP qualifier runs the NOKEEP script to delete the file provided you did not also specify the /RESERVE qualifier. If your VDE library does not have a NOKEEP script, the file is not deleted.

/LOG (default)
/NOLOG

Controls whether log messages are printed after each module is replaced. The /LOG qualifier causes the messages to be printed and the /NOLOG qualifier suppresses them. The messages indicate that each module has been replaced and that the database transaction has successfully committed.

/NOTE=note-number

The **note-number** parameter specifies the note number for the (previously-entered) SCT note associated with this replacement.

VDE can also enter the note into the conference directly, see the /SCT qualifier for information.

/OTHER="other-reason"

Specifies the type of check-in. The *other-reason* parameter specifies the reason for this check-in, and should indicate why this particular check-in was not classified as a fold check-in nor as an SCT check-in.

Also see /FOLD and /SCT.

/PROPAGATE=stream-name
/NOPROPAGATE

This qualifier controls the default module propogation behaviour of the replacement operation. The default for this qualifier can be altered by the propogation qualifier on the RESERVE command.

When specified, the /PROPAGATE qualifier causes VDE to propagate the changed module(s) only as far as the stream specified by the **stream-name** parameter, but *not* to any successors of that stream—change propagation thus stops at the specified stream.

The /NOPROPAGATE qualifier causes VDE to disable all propagate of the change beyond the current stream (the default stream or the stream specified with the /STREAM qualifier).

If you omit both qualifiers, the REPLACE command will propagate the changed modules to all successors of the current stream.

By default, neither /NOPROPOGATE nor /PROPOGATE is specified, and the module changes will propogate to all subsequent streams. The behaviour is affected by the specification of the propogation qualifier on the RESERVE command.

/QAR
/NOQAR

If you have enabled the library attribute that causes the REPLACE command to ask for the reason for the replacement, these qualifiers control whether VDE also asks for problem reports related to the replacement. The /QAR qualifier causes VDE to prompt for related QARs and CLDs (Digital problem reports) even if you give the reason as FOLD or OTHER. The /NOQAR qualifier causes VDE to not prompt for QAR and CLD information even if you give an SCT entry as the reason. If you specify neither /QAR nor /NOQAR, the REPLACE command prompts for QARs and CLDs only if you specify an SCT entry as the reason for the replacement. Neither qualifier has any effect if you have disabled the library attribute that causes the REPLACE command to ask for the reason for the replacement.

/QUEUE
/IMMEDIATE (default)

Controls whether each module is replaced into the specified stream immediately or is queued for later replacement. The /IMMEDIATE qualifier causes the REPLACE command to replace each module into the specified stream immediately so that the replacement is complete as soon as the command finishes. The /QUEUE qualifier causes the REPLACE command to queue the replacement in the VDE database—and causes a copy of the module to be placed into the staging area—so that the module remains reserved until an authorized user enters a PERFORM REPLACEMENT command to cause the actual replacement to occur. If any stream into which you are replacing the module has the **queued-replacement** attribute, the module is automatically queued for replacement and the /IMMEDIATE is ignored.

Of /QUEUE and /IMMEDIATE, /IMMEDIATE is the default.

/PROJECT=project-name

This qualifier allows the user to specify the project name this replacement is associated with. The project name can be used to track the work and changes related to a particular project.

This qualifier is part of the statistics collection component of VDE. Statistics collection is enabled or disabled on a library basis, using the MODIFY LIBRARY/STATISTICS command. If statistics collection is disabled, the statistics qualifiers are ignored and have no adverse effects. Certain keywords on this command *require* the specification of the /DEFECTS_CORRECTED or /NODEFECTS_CORRECTED qualifier.

/REASONS=reason-code

This qualifier allows the user a way to secify the reason the replacement is being made: **reason-code** specifies the particular reason code for this replacement, based on the list of reasons contained in Table 3–2.

Table 3–2 Reasons

Change Code	Defect Count Collection	Reason For Replacement
BF	Yes	The replacement is a Bug Fix
NF	No	New Feature or Enhancement
PI	No	Performance Improvement
CS	No	Common Sources
CU	Yes	Code Cleanup, Retrofit, or a fix for a Regression
NH	No	New Hardware Support
FE	No	Functional Equivalence
OT	No	Other

The **reason-code** is listed in Table 3–2, as is a brief explaination of the code, and an indication as to whether or not the number of defects is collected (see /DEFECTS_CORRECTED) for the particular reason code.

This qualifier is part of the statistics collection component of VDE. Statistics collection is enabled or disabled on a library basis, using the MODIFY LIBRARY/STATISTICS command. If statistics collection is disabled, the statistics qualifiers are ignored and have no adverse effects. Certain keywords on this command *require* the specification of the /DEFECTS_CORRECTED or /NODEFECTS_CORRECTED qualifier.

If statistics collection is enabled on the current library and you do not specify this qualifier, you will be prompted for the information.

/REMARK="string"

Specifies a character string to be associated with the newly created generation in the VDE database. The remark is enclosed in quotation marks. If no remark is entered, then the remark from the corresponding reservation is used.

The remark string is used as the title of the SCT note, if VDE was asked to enter the SCT note by the specification of the /SCT qualifier.

/REPLACEMENT=rep-name

Specifies that the reservations for the specified modules be added to a queued replacement with the name given by the **rep-name** parameter. If a queued replacement by that name already exists, the reservations are added to that replacement. If no such replacement exists, a new queued replacement by that name is created and the module reservations are added to it. This qualifier is ignored for immediate replacements; it is only meaningful if you also specify the /QUEUE qualifier or one of the streams into which you are replacing the modules have the **queued-replacement** attribute.

Queued replacements are implicitly and automatically created whenever a REPLACE is performed on a stream marked for queued replacements. For further information on explicitly creating a queued replacement, see the CREATE REPLACEMENT command.

/RESERVE
/NORESERVE (default)

Controls whether the replaced module is kept reserved. If you specify the /RESERVE qualifier, the module remains reserved and the input file is not deleted from your default directory. If you specify /NORESERVE, the module is no longer reserved after being replaced.

/REVIEWER=(username [,username...])

Specifies reviewers for the replacement. Each **username** parameter specifies the OpenVMS username of a user who should review all code replaced by the current command. The user must already be recorded in the VDE database. VDE notifies the user by mail message that the replacement has occurred. If the replacement is queued, the user should then review the changes in that replacement before it is performed. If the replacement is immediate, VDE notifies the user of the new module generations that were created, and the user may then inspect those generations.

If you specify only one **username** parameter, you may omit the surrounding parentheses.

/SCT=filename

The **filename** parameter specifies the name of a file to be loaded into the SCT notes conference. This file contains the previously-edited text of an SCT entry for the SCT note associated with this replacement.

If this qualifier is used, the default notes conference for the stream—SCT-*stream-name*—is assumed.

If this qualifier is specified, the /REMARK string is used as the title of the SCT note.

For information on the proper creation and protection of an SCT notes conference, see the *Guide to Using VDE*.

Also see /FOLD and /OTHER.

/SESSION=session-name

Sessions are used to logically group a set of module reservations together, typically to group all modules related to a particular source code alteration or enhancement together. It allows all component modules reserved to be treated as a single entity for subsequent replacement operations. A session also allows additional modules to be reserved and incorporated into an existing session at a later time.

If you use the /SESSION qualifier, those reservations that are members of the specified reservation session are replaced. In this case, you must omit the **mod-name** parameters from the command.

Sessions can be manipulated via the REPLACE, RESERVE, UNRESERVE, MODIFY SESSION, MODIFY RESERVATION, CREATE MODULE, and CANCEL SESSION commands. And modules created by CREATE MODULE (on a queued-replacement stream) and reserved via RESERVE can be combined into the same session.

/STREAM=stream-name

Specifies that the modules be replaced into the development stream given by the **stream-name** parameter. The changed modules are also propagated to any successors of that stream, if appropriate. If this qualifier is omitted, the modules are replaced into the default development stream. If this qualifier is omitted and no default stream is defined, VDE prompts you for the stream name.

/USERNAME=username

Specifies that modules be replaced on behalf of another user. The **username** parameter specifies the OpenVMS username of the other user. That user must have the module reservations that the replace operation will end. You must have the USERNAME privilege to use this qualifier.

EXAMPLES

1

```
VDE> REPLACE MOD1 /REMARK="Fixed list-update bug"
%VDE-I-FILCOPLIB, file DEV$:[MYDIR]MOD1.PAS;2 copied to library
%VDE-I-REPLACED, generation [FACIL]MOD1.PAS;8(7A1) replaced into stream V2.0
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example replaces the module MOD1.PAS into the current VDE library. Because no facility is specified with the command, the module is assumed to exist in the default facility, FACIL. Because no type name is specified, all source modules with name MOD1 are replaced; in this case there is only one such module, MOD1.PAS. The log messages show that the module was successfully replaced into stream V2.0.

2

```
VDE> REPLACE [COPY]COPYMAIN.B32
Please supply required information for module [COPY]COPYMAIN.B32:
Enter the replacement remark: ADD THE /FUMBLE QUALIFIER
%VDE-I-FILCOPLIB, file DEV$:[MYDIR]COPYMAIN.B32;3 copied to library
%VDE-I-REPLACED, generation [COPY]COPYMAIN.B32;23(23) replaced into stream V3.0
%VDE-I-REPLACED, generation [COPY]COPYMAIN.B32;23(23) replaced into stream V3.1
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This command replaces module COPYMAIN.B32 in facility COPY into the current library. Because the /REMARK qualifier is not specified with the command, VDE prompts for the remark before replacing the module. The log messages shows that the REPLACE command replaced the module into stream V3.0 and then propagated the new generation into stream V3.1.

3

```
VDE> REPLACE/QUEUE FEE.MAR, FOO.MAR "Remark string"
%VDE-I-FILCOPSTAG, file DEV$:[MYDIR]FEE.MAR;3 copied to staging area
%VDE-I-FILCOPSTAG, file DEV$:[MYDIR]FOO.MAR;3 copied to staging area
%VDE-I-REPLQUE, module [FACIL]FEE.MAR queued for replacement SMITH-1
%VDE-I-REPLQUE, module [FACIL]FOO.MAR queued for replacement SMITH-1
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

In this example, the REPLACE command queues the replacement of modules FEE.MAR and FOO.MAR in facility FACIL. The modules are copied to a staging area that VDE manages, but they are not replaced into the VDE library and they remain reserved. The REPLACE command gives the queued replacement the generated name SMITH–1. To actually

replaces the two modules, an authorized user must enter a PERFORM REPLACEMENT command for replacement SMITH–1.

RESERVE

Retrieves the latest generation of one or more source modules from a given development stream and marks those modules as reserved by the current user. Related reservations can be grouped and manipulated together using sessions.

Requires RESREP privilege.

FORMAT

RESERVE *mod-name [, mod-name...] ["remark"]*

PARAMETER

mod-name

Specifies a source module in the default stream or the stream specified with the /STREAM qualifier to be reserved from the library. The module name consists of an optional facility name enclosed in square brackets, a module name, and an optional type name preceded by a period (such as [FACIL]MOD1.MAR). If you omit the facility name, source modules from the default facility are reserved. If you omit the type name, all source modules with the given module name in the given facility are reserved.

You can reserve more than one module by using wildcard characters in any of the three components of the module name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. VDE reserves the source modules whose names match the wildcard pattern.

You can also reserve multiple modules by specifying the name of a source group instead of a module name. Source groups are created with the CREATE GROUP command. If you specify a group name, each module that is a member of the group is reserved from the library.

"remark"

Specifies a remark string for the reservation. Specifying the **remark** parameter is equivalent to specifying the /REMARK qualifier after the RESERVE keyword.

DESCRIPTION

The RESERVE command copies the latest generation of a module in a specified stream to your default directory (or to another location if you specified the /OUTPUT qualifier) and marks the module as ‘‘reserved’’ in that stream.

After you have modified the module, you can return your changes to the library with the REPLACE command. Alternatively, you can cancel the reservation with the UNRESERVE command; no module modifications will be entered into the library.

Note: The module reservation will be released with the UNRESERVE command, by the REPLACE command (when queued-replacement are not enabled on the stream), or by the PERFORM REPLACE command (when queued-replacement is enabled on the stream).

RESERVE

See the REPLACE, UNRESERVE, SHOW REPLACEMENT, and SHOW RESERVATION commands.

By default, VDE does not allow you to reserve a module that is already reserved by another user (or yourself) for the same development stream. However, you can reserve an already reserved module if you specify the /OVERRIDE=CONCURRENT qualifier and the already existing reservations all allow concurrent reservations. If you do not wish to override a reservation, you can use the /QUEUE qualifier to request notification when the module becomes available for reservation.

When you reserve a generation of a module from a VDE library, the file that is placed in your directory has the same creation and revision times, record format, and record attributes as the file that was used to create the generation that you are reserving.

To reserve modules on behalf of another user, use the /USERNAME qualifier to specify the OpenVMS username of that other user. The modules are then reserved in the specified user's name. You must have the USERNAME privilege to use the /USERNAME qualifier.

For information on creating a new module, see CREATE MODULE. Depending on the queued-replacement setting of the target stream, you may need to issue a REPLACE command after CREATE MODULE; see REPLACE. The behaviour of the CREATE MODULE and the RESERVE commands are identical, save for the initial absence of the module in the library for the CREATE MODULE, and the initial presence of the module in the library for the RESERVE.

On queued-replacement streams, modules created via a CREATE MODULE and modules reserved via a RESERVE can be combined into a single replacement, and can be combined into a single reservation /SESSION.

QUALIFIERS

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each module reserved. The /CONFIRM qualifier causes VDE to print a message for each module that you have specified asking whether you want to reserve the corresponding module generation. If you answer YES (or Y), it is reserved. If you answer NO (or N), it is not reserved. The /NOCONFIRM qualifier causes VDE to reserve each specified module without asking for confirmation.

/FOLD[=fold-ident]

Specifies a fold record to be cancelled when this module reservation is eventually replaced into the library. The **fold-ident** parameter gives the fold identifier of a fold record for the module being reserved and for the stream in which the reservation is done. When you later replace the module and VDE performs the replacement, VDE will cancel this fold record from the VDE database. You can omit the **fold-ident** parameter if there is only one fold record for the specified module and stream. You can use the /FOLD qualifier if you are modifying the module to satisfy an existing fold record. You can also specify the /FOLD qualifier with the REPLACE command to achieve the same effect.

/HISTORY["string"]
/NOHISTORY

The /HISTORY qualifier specifies that CMS history information is to be included in the file placed in your directory. The history of the module lists the generations that preceded the current generation of the module. Each generation produces one line of history information in the output file. This line gives the CMS generation number, user name, date, time, and remark associated with the generation.

The quoted string specifies the format of the history line. The quoted string must contain the characters #H or #B (lower case is allowed) and can contain other printing characters (such as comment delimiters). Each history line is formed by inserting the history information where you specify #H or #B in the history string. If you specify #H, the history information is inserted at the end of the file and if you specify #B it is inserted at the beginning of the file. To include a quotation mark in the output history string, type it twice (""). To include a number sign (#) in the output history string, type it twice (##). If the **string** parameter is omitted, VDE uses the module's default history string.

The /NOHISTORY qualifier specifies that history information is not to be appended to the output file placed in your directory.

If the /HISTORY and /NOHISTORY qualifiers are both omitted, the RESERVE command generates history information only if the history attribute is enabled for the module. This attribute is set with the CREATE MODULE or MODIFY MODULE command.

If the history-notes attribute for the VDE library is disabled, the RESERVE command will not generate CMS history information into the output file and will not accept the /HISTORY qualifier. See the description of the /HISTORY_NOTES qualifier to CREATE LIBRARY for more information.

/IDENTIFICATION=res-ident

Specifies a reservation identifier for the new reservation. Each reservation has a unique reservation identifier so that you can distinguish between multiple concurrent reservations of the same module by the same user. By default, the RESERVE command assigns a unique numeric identifier to each reservation, such as ''1'', ''2'', and so on. The /IDENTIFICATION qualifier overrides this default and allows you to select a reservation identifier of your choice, such as a mnemonic name. The **res-ident** parameter specifies the reservation identifier. It can be up to 39 characters long and must follow VDE name syntax.

/LOG (default)
/NOLOG

Controls whether log messages are printed after each module generation is reserved. The /LOG qualifier causes such messages to be printed and /NOLOG suppresses them. The messages indicate that the modules have been reserved and that the database transaction has successfully committed.

/MERGE=stream-name
/MERGE=(gen-expr)

Controls whether another generation of the module (called the merge generation) is to be merged with the generation that is being reserved

(called the retrieved generation). If you specify the /MERGE qualifier with the **stream-name** parameter, the merge generation is the latest generation of the module in the specified stream. If you specify the /MERGE qualifier with the **gen-expr** parameter enclosed in parentheses, the merge generation is the generation with the specified generation expression (for example, ''2A3'').

When you specify the /MERGE qualifier, VDE merges the lines of the retrieved generation with the lines of the merge generation and delivers the merged text as a single file to your output directory. The file that is placed in your directory has the current creation and revision times. The merge generation cannot be on the same line of descent as the retrieved generation. When there is a conflict between blocks of one or more lines, VDE includes the conflicting lines and flags the conflict.

VDE uses the callable interface to CMS to provide the merge capability. For a detailed explanation of how two generations are merged and how CMS (and therefore VDE) treats conflicts between the generations, see the CMS documentation.

/NOCONCURRENT

Specifies that the module cannot be reserved by another user while you have it reserved. You must replace or unreserve the module before others can reserve it. By default, VDE allows concurrent reservations if the module has the **concurrent** attribute set.

/NOTES[="string"]
/NONOTES

The /NOTES qualifier specifies that CMS notes are to be appended to the lines of the output file. CMS notes indicate in what module generation each source line was created.

The quoted string specifies the format of the note. The quoted string can contain text (including the appropriate comment delimiter) and the characters #G or #g. The CMS generation expression of the generation in which the line first appeared in inserted where you specify #G or #g. The resulting string is appended to the end of the output line. To include a quotation mark in the output notes string, type it twice (`""`). To include a number sign (#) in the output notes string, type it twice (##). If the **string** parameter is omitted, VDE uses the module's default notes string. (You set the default notes string with the CREATE MODULE or MODIFY MODULE command.)

The /NONOTES qualifier specifies that CMS notes are not to be appended to the lines of the file placed in your directory.

If the /NOTES and /NONOTES qualifiers are both omitted, the RESERVE command generates CMS notes only if the notes attribute is enabled for the module. This attribute is set with the CREATE MODULE or MODIFY MODULE command.

If the history-notes attribute for the VDE library is disabled, the RESERVE command will not generate CMS notes information into the output file and will not accept the /NOTES qualifier. See the description of the /HISTORY_NOTES qualifier to CREATE LIBRARY for more information.

/NOOUTPUT

Specifies that the reserve operation is to be performed along with any history processing and error checking but that no output file is to be created.

/OUTPUT=file-specification

Directs VDE to write the reserved module to the specified file. If you omit the directory from the file specification, VDE writes the file to your default directory. If you omit the file name or file type, VDE uses the module name and type by default. If you omit the /OUTPUT and /NOOUTPUT qualifiers entirely, VDE creates a file in your current default directory with the same name as the module.

If you reserve more than one module with the same /OUTPUT qualifier and you specify the file name and type on that qualifier, VDE creates successive versions of the file indicated by the /OUTPUT qualifier.

/OVERRIDE=CONCURRENT

Specifies that VDE should create a concurrent reservation if a module being reserved is already reserved for the same stream. If this qualifier is omitted and a module you are trying to reserve is already reserved, VDE displays an error message and does not reserve the module for you. To create a second, concurrent reservation, you must use this qualifier. Even when you use this qualifier, VDE does not allow you to create a concurrent reservation if a previous reservation for the module does not allow concurrent reservations.

/POSITION=column-number

Specifies the column in which CMS notes are to be placed. The column number must be an integer in the range 1 to 511. If the length of an output line is less than the specified column number, the note appears at the specified column number. If the length of the line is greater than or equal to the column number, the note is placed at the next tab stop after the end of the line. (Tab stops are at position 9 and every 8 characters thereafter.)

If the /POSITION qualifier is omitted, the default notes position for the module is used. The default notes position is set with the CREATE MODULE or MODIFY MODULE command. The /POSITION qualifier has no effect unless the /NOTES qualifier is specified or the notes attribute is set for the module.

If the history-notes attribute for the VDE library is disabled, the RESERVE command will not generate CMS notes information into the output file and will not accept the /POSITION qualifier. See the description of the /HISTORY_NOTES qualifier to CREATE LIBRARY for more information.

/PROPAGATE=stream-name
/NOPROPAGATE

This qualifier controls the default module propogation behaviour of the subsequent and associated REPLACE command, and it controls the reservation ‘‘shadow’’ for this particular reservation.

When specified, the /PROPAGATE qualifier causes VDE to propagate the changed module(s) only as far as the stream specified by the **stream-name** parameter, but *not* to any successors of that stream—change propagation thus stops at the specified stream.

The /NOPROPAGATE qualifier causes VDE to disable all propagate of the change beyond the current stream (the default stream or the stream specified with the /STREAM qualifier).

If you omit both qualifiers, the REPLACE command will propagate the changed modules to all successors of the current stream.

Because these qualifiers determine what streams a subsequent REPLACE command will affect, they also determine what streams—the ''shadow''—the new module reservations cover. A reservation that covers fewer streams allows other users to reserve the same module in other streams with less likelihood of conflict.

For information on propogation, see REPLACE.

/QUEUE

Specifies that VDE should queue a reservation request for you if you cannot reserve a module because it is already reserved by another user. If the requested module can be reserved now, the module is reserved and this qualifier has no effect. However, if the module is not available, VDE queues a reservation request for it in its database. When the user who owns the current reservation later replaces or unreserves it, VDE sends you a mail message saying that the module is now available. To actually reserve the module, you must then enter another RESERVE command. If multiple users have queued reservation requests for the same module, all of them are notified when the module becomes available. They must then decide among themselves who should reserve the module first.

To display queued reservation requests, use the SHOW RESERVATION command.

/REMARK="string"

Specifies a character string to be stored in the database for this reservation, usually explaining why the reservation is made. The remark is enclosed in quotation marks. If no remark is entered, a null remark is stored.

/SESSION[=session-name]

Sessions are used to logically group a set of module reservations together, typically to group all modules related to a particular source code alteration or enhancement together. It allows all component modules reserved to be treated as a single entity for subsequent replacement operations. A session also allows additional modules to be reserved and incorporated into an existing session at a later time.

If no session with the specified **session-name** session name currently exists, VDE will create a new reservation session under the specified session name and will incorporate the module reservation(s) into it.

If the **session-name** parameter is omitted, VDE generates a unique session name, creates a new session by that name, and adds the module reservation to it.

If the /SESSION qualifier is omitted on the RESERVE command, VDE will reserve the module (as expected), and will not add the new module reservation to any session.

Sessions can be manipulated via the REPLACE, RESERVE, UNRESERVE, MODIFY SESSION, MODIFY RESERVATION, CREATE MODULE, and CANCEL SESSION commands. And modules created by CREATE MODULE (on a queued-replacement stream) and reserved via RESERVE can be combined into the same session.

/STREAM=stream-name

Specifies that modules be reserved in the development stream given by the **stream-name** parameter. The RESERVE command fetches the latest generation of each specified module in that stream and records that each reservation belongs to that stream. If this qualifier is omitted, modules are reserved in the default development stream. If this qualifier is omitted and no default stream is defined, VDE prompts you for the stream name.

/USERNAME=username

Reserves modules in another user's name. The **username** parameter specifies the OpenVMS username of the other user. You must have the USERNAME privilege to use this qualifier.

EXAMPLES

1

```
VDE> RESERVE FOO/REMARK="Fix problem reported in QAR 23"
%VDE-I-FETCHED, generation [FACIL]FOO.MAR;7(5A2) fetched from library
%VDE-I-OUTFILE, output file is DEV$:[MYDIR]FOO.MAR;
%VDE-I-RESERVED, module generation [FACIL]FOO.MAR;7(5A2) reserved from stream MAIN
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This command reserves all source modules named FOO in the default facility, FACIL. In this case, there is only one such module, FOO.MAR. This module is copied to the user's default directory and is marked as reserved in the VDE database.

2

```
VDE> RESERVE MOD1 /IDENTIFICATION=BUGFIX
Please supply required information for module [FACIL]MOD1.*;0:
Enter the reservation remark: FIX ACCESS VIOLATION
%VDE-I-FETCHED, generation [FACIL]MOD1.PAS;9(9) fetched from library
%VDE-I-OUTFILE, output file is DEV$:[MYDIR]MOD1.PAS;
%VDE-I-RESERVED, module generation [FACIL]MOD1.PAS;9(9) reserved from stream V6.0
%VDE-I-COMMIT, database transaction has successfully committed
VDE> SHOW RESERVATION
Reservation BUGFIX of [FACIL]MOD1.PAS       SMITH        29-DEC-1989 V6.0
VDE>
```

The RESERVE command reserves module MOD1.PAS in facility FACIL and assigns the reservation identifier BUGFIX to the reservation. (If you do not use the /IDENTIFICATION qualifier, the RESERVE command assigns a numeric reservation identifier to the reservation.) The SHOW RESERVATION command indicates that the module is reserved with the BUGFIX identifier. This identifier can later be used on a REPLACE or UNRESERVE command to identify this particular reservation.

3
```
VDE> RESERVE/SESSION=SPEEDUP A.REQ,B.REQ "Remark string"
%VDE-I-FETCHED, generation [FACIL]A.REQ;11(11) fetched from library
%VDE-I-OUTFILE, output file is DEV$:[MYDIR]A.REQ;
%VDE-I-FETCHED, generation [FACIL]B.REQ;7(7) fetched from library
%VDE-I-OUTFILE, output file is DEV$:[MYDIR]B.REQ;
%VDE-I-RESERVED, module generation [FACIL]A.REQ;11(11) reserved from stream MAIN
%VDE-I-RESERVED, module generation [FACIL]B.REQ;7(7) reserved from stream MAIN
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example shows how module reservations are made members of a reservation session. This command reserves modules A.REQ and B.REQ in facility FACIL and makes the resulting reservations members of a reservation session called SPEEDUP. These modules can later be replaced (or unreserved) as a unit by specifying the session name instead of the names of the individual modules.

REVIEW BUILD_JOB

Reviews the build-step log files for a specified build job.

FORMAT

REVIEW BUILD_JOB *[stream-name]*

PARAMETER

stream-name

The name of a development stream. VDE reviews the results of the most recent build job for the specified stream. If you omit the **stream-name** parameter, VDE reviews the results of the most recent build job for the default stream.

DESCRIPTION

The REVIEW BUILD_JOB command allows you to review the log files of the individual build steps of a specified build job. The command enters you into a subsystem of VDE. Once in the subsystem, you are prompted with the string VDEREVBLD> and you can then use a restricted set of commands, different from usual VDE commands, to display selected log files. After reviewing build results, use the EXIT command to exit from the subsystem and return to VDE.

The REVIEW BUILD_JOB command creates a *build-step review list* which is a list of build steps to be reviewed. By default, the list includes only those steps that completed with errors. Commands in the REVIEW BUILD_JOB subsystem list the steps in the review list and display the log files for those steps. The subsystem thus gives you a convenient way to determine which build steps failed and why they failed. The subsystem can also review build steps that completed successfully. Use the /STEP qualifier to include successful steps in the build-step review list.

When you are in the REVIEW BUILD_JOB subsystem, you enter commands at the VDEREVBLD> prompt. The commands can display either the build-step review list or the log files for individual build steps on that list.

The REVIEW BUILD_JOB subsystem accepts the commands documented in Review Build Job Commands.

QUALIFIERS

/STEP [=(status [, status...])]

Reviews either all build steps or, if you specify the **status** parameter, those steps whose status is specified. Each **status** parameter must be one of the following keywords: NOTSTARTED, RUNNING, SUCCESS, or ERRORS.

If you specify only one **status** parameter, the parentheses may be omitted. If you do not specify a parameter, all build steps of specified build job are reviewed.

If you omit the /STEP qualifier, only those build steps that completed with errors are reviewed.

EXAMPLES

1

```
VDE> REVIEW BUILD_JOB
Reviewing steps of build job 38 for stream MAIN
Number of build steps selected: 1

VDEREVBLD> DIRECTORY
Steps being reviewed in build job 38 for stream MAIN:
   Compile step for module [BUILD_TEST]E.B32

VDEREVBLD> NEXT
Build job 38 for stream MAIN
Log file of compile step for module [BUILD_TEST]E.B32:

     1: --->> ERROR: MISSING SCRIPT IN VDE DATABASE <<---
     2: No script found in database to perform build step
[End of file]

VDEREVBLD> NEXT
%VDE-E-NOMORESTEP, no more build steps in review list
VDEREVBLD> EXIT
VDE>
```

In this example, one step completed with errors in build job 38 for stream MAIN. The REVIEW BUILD_JOB command displays messages that identify the build job and gives the number of steps selected for review. By default, the command includes only those steps from the build-step review list that completed with errors; in this case, there is one such step. At the VDEREVBLD> prompt, the user enters the DIRECTORY command to display the build steps on the current review list. Then the user enters the NEXT command to see the log file for the next build step. (Pressing the RETURN key has the same effect.) The log file shows that the compile step failed because no script for it was found in the VDE database. A second NEXT command shows that there are no more build steps on the review list. The EXIT command then exits from the REVIEW BUILD_JOB subsystem and returns the user to the VDE> prompt.

2

```
VDE> REVIEW BUILD_JOB/STEP=SUCCESS
Reviewing steps of build job 39 for stream MAIN
Number of build steps selected: 5

VDEREVBLD> DIRECTORY
Steps being reviewed in build job 39 for stream MAIN:
   Compile step for module [BUILD_TEST]B.REQ
   Compile step for module [BUILD_TEST]C.B32
   Compile step for module [BUILD_TEST]D.B32
   Compile step for module [BUILD_TEST]E.B32
   Link step for module [BUILD_TEST]X.EXE

VDEREVBLD> SELECT X
Build job 39 for stream MAIN
Log file of link step for module [BUILD_TEST]X.EXE:

     1: $ DEFINE OBJLIB DISK$:[LIBROOT.MAIN.BUILD_TEST.OBJ]
     2: $ LINK/EXE=OBJLIB:X.EXE/MAP=OBJLIB:X.MAP  -
     3:         OBJLIB:OBJLIB.OLB/INCLUDE=C,OBJLIB:OBJLIB.OLB/LIBRARY
     4: $ DEASSIGN OBJLIB
     5: $ !
     6: $ SET NOVERIFY   ! Successful script completion
[End of file]

VDEREVBLD> EXIT
VDE>
```

In this example, the /STEP=SUCCESS qualifier causes the REVIEW BUILD_JOB command to include only those steps from the build-step review list that completed successfully. The subsystem DIRECTORY command then lists all such steps. The SELECT command displays the log file of the link step for module X.EXE. Finally, the EXIT command exits from the REVIEW BUILD_JOB subsystem and returns the user to the VDE> prompt.

Review Build Job Commands

The Review Build Job commands are used to review the results of a build job. The commands are valid at the VDEREVBLD> prompt during the REVIEW BUILD_JOB command.

FORMAT

DIRECTORY *[mod-name [, mod-name...]]*
SELECT *[mod-name]*
NEXT *[STEP]*
BACK
CURRENT
FIRST
LAST
SPAWN
ATTACH
EXIT

DESCRIPTION

The review commands allow you to review the log files of the individual build steps of a specified build job. The command enters you into a subsystem of VDE. Once in the subsystem, you are prompted with the string VDEREVBLD> and you can then use a restricted set of commands, different from usual VDE commands, to display selected log files. After reviewing build results, use the EXIT command to exit from the subsystem and return to VDE.

The REVIEW BUILD_JOB command creates a *build-step review list* which is a list of build steps to be reviewed. By default, the list includes only those steps that completed with errors. Commands in the REVIEW BUILD_JOB subsystem list the steps in the review list and display the log files for those steps. The subsystem thus gives you a convenient way to determine which build steps failed and why they failed. The subsystem can also review build steps that completed successfully. Use the /STEP qualifier to include successful steps in the build-step review list.

When you are in the REVIEW BUILD_JOB subsystem, you enter commands at the VDEREVBLD> prompt. The commands can display either the build-step review list or the log files for individual build steps on that list. The subsystem accepts the following commands:

- DIRECTORY *[mod-name [, mod-name...]]*

  Displays a list of all build steps on the build-step review list, giving the type of each step (compile, link, or copy) and the name of the module that is compiled, linked, or copied. If you specify one or more **mod-name** parameters, the command displays only those build steps whose module names match the parameters. You can use wildcards in the **mod-name** parameter.

- SELECT *[mod-name]*

  Selects the first build step whose module name matches the **mod-name** parameter and displays the log file for that build step. You can use wildcards in the **mod-name** parameter.

  If you omit the **mod-name** parameter, the command displays the log file of the next build step on the review list that matches the most recently specified module name from a previous SELECT command. Using this feature, you can enter a SELECT command with a **mod-name** parameter containing wildcards to display the log file of the first build step that matches the parameter. You can then enter SELECT commands without parameters to display the log files of all other build steps that match the original parameter, one log file per SELECT command.

- NEXT [STEP]

  Displays the next page of the log file of the current build step. If you specify the optional STEP keyword, the first page of the log file of the next build-step is displayed.

  Pressing the KP0 key (the 0 key on the numeric keypad) is the same as entering the NEXT command. As a result, you can review the log files of all build steps on the review list by repeatedly pressing the KP0 key.

- BACK

  Displays the log file of the previous build step on the build-step review list.

- CURRENT

  Displays the first page of the log file of the current build step.

- FIRST

  Displays the log file of the first build step on the build-step review list.

- LAST

  Displays the log file of the last build step in the build-step review list.

- SPAWN

  Spawns a subprocess and attaches the terminal to that subprocess. See the description of the VDE SPAWN command for details.

- ATTACH

  Attaches the terminal to a specified process. See the description of the VDE ATTACH command for details.

- EXIT

  Exits from the REVIEW BUILD_JOB subsystem and returns you to the VDE> prompt.

REVIEW REPLACEMENT

Reviews the modules, comment files, information file, and log file associated with a specified queued replacement.

FORMAT

REVIEW REPLACEMENT *rep-name*

PARAMETER

rep-name

The name of the queued replacement to review. This parameter is required; if you omit it, VDE prompts you for it.

DESCRIPTION

The REVIEW REPLACEMENT command allows you to review all files associated with a given queued replacement. These files include the modules being replaced, the information file that the replacing user may have provided, the comment files that other reviewers may have written, and possibly a log file for an attempt to perform the replacement. The log file is only present if an earlier attempt to perform the replacement failed; the log file then tells you wnat went wrong. For each module being replaced, you may examine the new version queued for replacement, the old version in the VDE library, and the differences between the two.

The REVIEW REPLACEMENT command enters you into a subsystem of VDE. Once in the subsystem, you are prompted with the string VDEREVREP> and you can then use a restricted set of commands, different from the usual VDE commands, to display the files associated with the replacement. After reviewing these files, you must use the EXIT command to exit from the subsystem and return to VDE.

The REVIEW REPLACEMENT command creates a list of files to be reviewed called the *replacement review list*. This list includes the replacement's information file (if any), the comment file for each reviewer that has provided one, the log file for the last attempt to perform replacement (if any), and the modules that are queued for replacement. Subsystem commands list, display, and navigate among the elements in the review list. One element is always designated as the *current element*. Initially, the first element on the list is the current element, but various command can change that designation. The SHOW and EDIT subcommands display the text of the current element.

QUALIFIERS

/COMMENT

Specifies that reviewer comment files be included in the replacement review list. If this qualifier is omitted, comment files are only included if all other qualifiers are omitted as well.

/INFORMATION

Specifies that the queuing user's information file be included in the replacement review list. If this qualifier is omitted, the information file is only included if all other qualifiers are omitted as well.

/MODULE

Specifies that all modules that are members of the queued replacement be included in the replacement review list. If this qualifier is omitted, modules are only included if all other qualifiers are omitted as well.

/STREAM=stream-name

This qualifier specifies the name of the stream associated with the replacement. It is currently ignored.

EXAMPLES

1

```
VDE> REVIEW REPLACEMENT SMITH-1
Reviewing replacement SMITH-1 queued by user SMITH (Jane Smith)
Number of modules and files selected: 4

VDEREVREP> DIRECTORY
Modules and files being reviewed for replacement SMITH-1:
   Information file by user SMITH (Jane Smith)
   Comment file by reviewer BROWN (Bill Brown) rejecting replacement
   Module [FACIL]A.REQ
   Module [FACIL]C.B32

VDEREVREP> CURRENT

Information file by user SMITH (Jane Smith)

VDEREVREP>
```

In this example, the REVIEW REPLACEMENT command displays some information identifying the requested queued replacement and enters the REVIEW REPLACEMENT subsystem. The DIRECTORY subcommand displays the elements on the review list and the CURRENT subcommand prints a line identifying the current element of the review list (namely the first element).

2

```
VDEREVREP> SHOW

Text of the queuing user's information file displayed here.
This file may consist of many lines.

VDEREVREP> NEXT

Comment file by reviewer BROWN (Bill Brown) rejecting replacement

VDEREVREP> SHOW

Text of comment file written by reviewer BROWN displayed here.
This file may consist of many lines.

VDEREVREP>
```

This example continues the previous example. The SHOW command displays the text of the current review-list element, user SMITH's information file. The NEXT command makes the next element the current

element and prints its name. The subsequent SHOW command then displays the text of that file, the comment file by reviewer BROWN.

```
3   VDEREVREP> NEXT

    Module [FACIL]A.REQ

    VDEREVREP> SHOW
    %VDE-I-FETCHED, generation [FACIL]A.REQ;7(7) fetched from library
    %VDE-I-OUTFILE, output file is DEV:[VDELIB.VDE$STAGE.VDE$STG_0.VDE$REP_11.
    FACIL]A.REQ_VDE$OLD;

    Number of difference sections found: 0
    Number of difference records found: 0

    DIFFERENCES /IGNORE=()/MERGED=1/OUTPUT=DEV:[VDELIB.VDE$STAGE.VDE$STG_0.
    VDE$REP_11.FACIL]A.REQ_VDE$DIFF;1-
        DEV:[VDELIB.VDE$STAGE.VDE$STG_0.VDE$REP_11.FACIL]A.REQ;1-
        DEV:[VDELIB.VDE$STAGE.VDE$STG_0.VDE$REP_11.FACIL]A.REQ_VDE$OLD;1

    VDEREVREP> NEXT

    Module [FACIL]C.B32

    VDEREVREP> NEXT
    %VDE-E-NOMOREMOD, no more modules in review list
    VDEREVREP> EXIT
    VDE>
```

Yet another NEXT command makes module [FACIL]A.REQ the current element of the review list. The SHOW command displays the differences file for that module. To do so, it first fetches out the old generation of the module, differences it with the new generation queued for replacement, and displays the differences file. The EDIT command could have been used just as well, and is more suitable for long differences files. Another NEXT command make module [FACIL]C.B32 the current element and displays its name. A final NEXT command shows that there are no more elements in the review list. The user then enters the EXIT command to return to the VDE> prompt.

Review Replacement Commands

The Review Replacement Commands are used to review replacements queued to the current library. The commands are valid at the VDEREVREP> prompt during the REVIEW REPLACEMENT command.

FORMAT

DIRECTORY *[mod-name [, mod-name...]]*
SELECT *[mod-name]*
SHOW *[keyword]*
EDIT *[keyword]*
NEXT
BACK
CURRENT
FIRST
LAST
ACCEPT *["remark-string"]*
REJECT *"remark-string"*
SPAWN
ATTACH
EXIT

DESCRIPTION

The REVIEW REPLACEMENT subsystem accepts the following commands at the VDEREVREP> prompt:

- DIRECTORY *[mod-name [, mod-name...]]*

  Displays a list of all information files, comment files, replacement log files, and modules on the replacement review list. For each file, this command displays the type (information, comment, log, module, or binary module) and the name of user who created the file (for information and comment files) or the name of the module. If you specify one or more **mod-name** parameters, the command displays only those modules whose names match the parameters. You can use wildcards in the **mod-name** parameter. An information or comment file has the author's OpenVMS username as its file name, so these files can also be selected with **mod-name** parameters. The replacement log file (if present) has the file name LOG.VDE$LOG.

- SELECT *[mod-name]*

  Selects the first review list element whose module or file name matches the **mod-name** parameter and makes that the current review list element. This command prints one line identifying the selected module or file. You can select an information or comment file by specifying the author's OpenVMS username in the **mod-name** parameter. You can

select the log file (if present) by specifying LOG in the **mod-name** parameter.

If you omit the **mod-name** parameter, the command displays the next element on the review list that matches the most recently specified module name from a previous SELECT command. Using this feature, you can enter a SELECT command with a **mod-name** parameter containing wildcards to select the first module (or other file) that matches the parameter. You can then enter SELECT commands without parameters to successively select all other modules and files that match the original parameter, one module or file per SELECT command.

- SHOW *[keyword]*

  Prints the text of the current review-list element on your terminal. If the current element is an information or comment file, VDE reads that file and prints it on your terminal. If the current element is a module, VDE prints the new version of the module (the version about to be replaced into the library), the old version of the module (the generation that was fetched when the module was reserved), or the differences between the two. The **keyword** parameter determines what is displayed for a module; this parameter may be the keyword NEW, OLD, or DIFFERENCES. This parameter is ignored for information, comment, and log files.

  If you specify the OLD keyword, VDE fetches out the old module generation unless a previous reviewer has already fetched it into the replacement's staging area. VDE then displays that file. If you specify the DIFFERENCES keyword, VDE fetches out the old module generation if necessary. VDE then runs the OpenVMS DIFFERENCES command to create a differences file unless a previous reviewer has already created it. VDE then displays that file.

  This command will not display any files for binary source modules.

- EDIT *[keyword]*

  Invokes your preferred editor in read-only mode to display the text of the current review-list element. If the current element is an information or comment file, VDE spawns the editor to display that file. If the current element is a module, VDE spawns the editor to display the new version of the module (the version about to be replaced into the library), the old version of the module (the generation that was fetched when the module was reserved), or the differences between the two. The **keyword** parameter determines what is displayed for a module; this parameter may be the keyword NEW, OLD, or DIFFERENCES. This parameter is ignored for information, comment, and log files.

  If you specify the OLD keyword, VDE fetches out the old module generation unless a previous reviewer has already fetched it into the replacement's staging area. VDE then displays that file. If you specify the DIFFERENCES keyword, VDE fetches out the old module generation if necessary. VDE then runs the OpenVMS DIFFERENCES command to create a differences file unless a previous reviewer has already created it. VDE then displays that file.

This command will not display any files for binary source modules.

Use the SET EDITOR command at the VDE> prompt to specify your preferred editor. If you do not declare an editor, VDE invokes the system default editor.

- NEXT

  Makes the next element on the review list the current element and displays the name of that element.

  Pressing the KP0 key (the 0 key on the numeric keypad) is the same as entering the NEXT command. As a result, you can step from element to element by repeatedly pressing the KP0 key. Similarly, the COMMA (,) key on the numeric keypad enters the NEXT and SHOW commands with a single keystroke and the MINUS (–) key on the numeric keypad enters the NEXT and EDIT commands.

- BACK

  Makes the previous element on the review list the current element and displays the name of that element.

- CURRENT

  Displays the name of the current element of the replacement review list.

- FIRST

  Makes the first element on the review list the current element and displays the name of that element.

- LAST

  Makes the last element on the review list the current element and displays the name of that element.

- ACCEPT *["remark-string"]*

  Records in the VDE database that you accept the replacement currently being reviewed. You must be a reviewer for the replacement to use this command. You can specify the optional remark string to give a reason for accepting the replacement. This command accepts the same qualifiers as the VDE-level ACCEPT REPLACEMENT command.

- REJECT *"remark-string"*

  Records in the VDE database that you reject the replacement currently being reviewed. You must be a reviewer for the replacement to use this command. The remark string should specify a reason for rejecting the replacement. If you omit the remark parameter, VDE prompts you for a remark. This command accepts the same qualifiers as the VDE-level REJECT REPLACEMENT command. Like REJECT REPLACEMENT, the REJECT command lets you specify a comment file to more fully explain your reasons for rejecting the replacement.

- SPAWN

  Spawns a subprocess and attaches the terminal to that subprocess. See the description of the VDE SPAWN command for details.

- ATTACH

  Attaches the terminal to a specified process. See the description of the VDE ATTACH command for details.

- EXIT

  Exits from the REVIEW REPLACEMENT subsystem and returns you to the VDE> prompt.

Script Keywords

The following keywords are used in VDE build scripts stored in the VDE database.

Requires CRESCRP privilege.

FORMAT

EMIT
EXIT
FETCH

DESCRIPTION

Scripts are used to add various user-defined extensions to various VDE functions and commands. VDE invokes various scripts at various times during VDE processing—see Script Types for further information on this.

Scripts of various types are recorded in the VDE database. A **script** is a sequence of keywords in the VDE script language that VDE can execute to create a DCL command file. VDE can then invoke that DCL command file in a subprocess to perform the action of the script. VDE uses scripts to perform many different kinds of actions. For example, VDE uses ''compile scripts'' to compile modules in a build job. Similarly, it uses ''new-stream scripts'' to populate the directories of a new stream with files when it performs the CREATE STREAM command. There are many other kinds of scripts, all distinguished by the keywords or circumstances that cause them to be invoked.

The primary purpose of VDE scripts is to insert names and directory specifications from the VDE database into text you specify to generate DCL command files that perform various actions. A VDE script is thus a mixture of plain text and invocations of functions that retrieve names (such as module names, type names, and facility names) and directory strings (such as the names of facility subdirectories) from the VDE database.

Since you define scripts, you define how VDE should perform the corresponding actions. For example, you decide how a Pascal module is compiled because you define the Pascal compilation script. Similarly, you can define a script that defines auxiliary actions of the REPLACE command. Scripts thus let you make policy decisions for your own project, while VDE provides the mechanisms that implement your policies.

When a new development stream is created, all script definitions from the parent stream are copied to the child stream. If modified, the script definitions then evolve independently in the two development streams.

A VDE script is a sequence of text lines that contain VDE script keywords. There are only three VDE script commands: EMIT, FETCH, and EXIT.

In addition, a script may contain blank lines and comment lines. The exclamation point (!) is the comment character. VDE ignores any text on a script line after an exclamation point, provided the exclamation point is not within quotes. A script keyword command may be continued from one line to the next by ending the first line with the sequence space-minus (–).

VDE accepts the following keyword commands in a script:

- EMIT *string [, string...]*

  Emits one line into the DCL command file. This keyword command evaluates each **string** parameter, concatenates the resulting character strings into a single line of text, and writes this line of text to the DCL command file being generated. Each **string** parameter may be a quoted character string, an invocation of a script function, or a directory symbol name.

  If the EMIT command parameter contains spaces or other characters, the string must be enclosed in a pair of quoting characters. (eg: If the command is a typical DCL command and contains blanks and various other characters, it must be quoted.) Valid quoting character pairs include the double-quote pair: (") and ("), and the double-caret pair: («) and (»). Use of the end-quoting character within the string is permissible, but requires that the character be doubled. (eg: a double-quote character within the string is specified by double double-quotes; by two consecutive double-quotes: ("").)

  If the parameter is an invocation of a script function, you must specify the function name or the function name followed by a parenthesized list of function parameters. VDE evaluates the function to produce a character string that is concatenated to the current DCL command line. (Next section lists the available script functions.)

  If the parameter is a directory symbol name, VDE looks up the OpenVMS directory specification associated with the directory symbol from the VDE database and concatenates that string to the current DCL command line. OpenVMS directory symbols are defined using the SET DIRECTORY command.

  To exit from the generated DCL in the script, issue one of the following script commands:

  ```
  EMIT «$ GOTO VDE$ERROR»
  EMIT «$ GOTO VDE$SUCCESS»
  EMIT "$ EXIT"
  ```

- EXIT

  Exits from the script during script execution. Subsequent script commands are thus not executed to produce DCL commands. (Note that the EXIT keyword does not create an EXIT command in the DCL command file; it only stops script execution. See the documentation on the EMIT script command for information on generating the proper DCL exit sequence from within a script.)

- FETCH *string [, string...]*

Fetches a module generation from the VDE library. This keyword evaluates each **string** parameter, concatenates the resulting character strings into a single line of text, and interprets the line as the qualifiers and parameters to a VDE FETCH command. Each **string** parameter may be a quoted character string, an invocation of a script function, or a directory symbol name.

Your **string** parameters should generate the full VDE FETCH command you want performed except for the FETCH keyword. The **string** parameters can generate all qualifiers and parameters that the VDE FETCH keyword accepts, and can use script functions to include strings such as module names, stream names, and directory specifications. The resulting command string is parsed and executed as if it belonged to a normal FETCH keyword entered from the terminal. Use the FETCH keyword when you want VDE to fetch a generation directly, without using a DCL command file, and you want it performed in the same database transaction as the operation performing the script.

By default, VDE fetches the latest generation of the specified module in the stream for which the script is invoked. You can override this default by generating a /STREAM qualifier, /GENERATION qualifier, or a generation number for the FETCH keyword. The script FETCH keyword suppresses log messages by default, but you can override this default by generating a /LOG qualifier. Otherwise, all parameters, qualifiers, and defaults are the same as for the regular VDE FETCH keyword described elsewhere.

If a **string** parameter is a quoted character string, the string must be enclosed in double quotes ("). A double-quote character within the string is specified by two consecutive double quotes (""). VDE concatenates the quoted string to the current FETCH keyword line.

If the parameter is an invocation of a script function, you must specify the function name or the function name followed by a parenthesized list of function parameters. VDE evaluates the function to produce a character string that is concatenated to the current FETCH keyword line. (Next section lists the available script functions.)

If the parameter is a directory symbol name, VDE looks up the OpenVMS directory specification associated with the directory symbol from the VDE database and concatenates that string to the current FETCH keyword line. OpenVMS directory symbols are defined using the SET DIRECTORY command.

When executing a script, VDE executes successive script keywords from the script until it encounters an EXIT keyword or the end of the script. VDE executes FETCH keywords immediately as they are encountered. VDE writes all lines generated by EMIT keywords to the output DCL command file. At the end of the script, VDE executes that command file in a subprocess to perform the action of the script.

Selection of Scripts

Many types of scripts are invoked when a certain operation must be applied to a specific module, such as compiling or reserving the module. For such operations, VDE must select the script that is appropriate to the module. The type of the script is of course the primary selection criterion, but even for a given type, different scripts are needed for different modules. The script that compiles a Fortran module is not appropriate for a C module, for example, and the script that links one image cannot link another. (Script Functions contains the types of scripts supported by VDE.)

To accommodate such situations, VDE lets you specify how scripts are selected based on the module name. You can specify that a script should be used only for a specific module, for all modules of a certain type in a given facility, for all modules of a given type regardless of facility, or for all modules regardless of type or facility. When you create such a script, you specify the script type with a qualifier, such as /COMPILE or /RESERVE. You also specify the modules to which the script applies through a module-name parameter to the qualifier. A restricted form of wildcarding determines what modules the script can be used for. The following module-name specifications are allowed:

- [FACIL]MOD.TYP

  If you fully specify the module name, the script applies only to that specific module. For example, the qualifier /LINK=[CODE]X.EXE says that the script can only be used to link image X.EXE in facility CODE. If you omit the facility name, the current default facility is assumed.

- [FACIL]*.TYP

  If you specify the facility name and type name but specify an asterisk (*) in place of the module name, the script applies to all modules of that type in that facility. For example, the qualifier /COMPILE=[FACIL]*.C says that the script can be used to compile all C modules in facility FACIL. If you omit the facility name, the current default facility is assumed.

- [*]*.TYP

  If you specify the type name but specify an asterisk (*) in place of both the facility name and the module name, the script applies to all modules of that type in all facilities. For example, the qualifier /COMPILE=[*]*.FOR says that the script can be used to compile all Fortran modules in all facilities.

- [*]*.*

  If you specify an asterisk (*) for each of the three components of the module name, the script applies to all modules regardless of type and facility. For example, the qualifier /RESERVE=[*]*.* says that the script can be used to reserve all modules in all facilities.

No other forms of wildcarding are allowed in the module-name parameters to script-type qualifiers.

When VDE looks for a script to perform some action for a module, it always selects the script that is most specific to that module. If it can find a script with a fully specified module name ([FACIL]MOD.TYP) for the module in question, it uses that script. Otherwise, it looks for a script with a matching module-name specification of the form [FACIL]*.TYP or [*]*.TYP or [*]*.*, in that order. This rule allows you to specify generic scripts that apply to all modules or to all modules of a certain type, but to override those scripts for specific modules or for modules in specific facilities when appropriate.

Similar rules apply to scripts that are selected by facility. A NEW_ STREAM script, for example, is invoked for each facility in your system when you create a new stream. In this case, VDE accepts two forms of facility-name specification on the script-type qualifiers:

- FACIL

  If you fully specify the facility name, the script applies only to that specific facility. For example, the qualifier /NEW_STREAM=RMS says that the script can only be used for facility RMS when VDE creates new streams.

- *

  If you specify an asterisk (*) for the facility name, the script applies to all facilities. For example, the qualifier /NEW_STREAM=* says that the script can be invoked for every facility in your system when VDE creates new streams.

No other forms of wildcarding are allowed in the facility-name parameters to script-type qualifiers.

When VDE looks for a script to perform an action for a facility, it first looks for a script with a fully specified facility name. If no such script exists in the database, VDE selects the script with the wildcard in the facility name, provided it exists. You can thus specify a generic script that applies to all facilities, and then override that script for specific facilities when appropriate.

Generating Command Files in Segments

Normally, VDE generates the DCL command file to perform some action from a single VDE script. In some situations, however, it is necessary to generate different ‘‘segments’’ of the DCL command file from different scripts. The DCL command file that compiles a MACRO module, for example, may need to consist of three segments:

- The first segment sets up logical names for the MACRO assembler and includes the MACRO command and its qualifiers. The MACRO command is ended with the continuation character. This segment is the same for all MACRO modules, and the VDE script that generates it has the selection template [*]*.MAR.
- The second segment contains the name of the module to assemble and a plus-list of macro libraries to use with that module. This segment of the command file is different for each MACRO module, and the

VDE script that generates it has a selection template of the form [FAC]MOD.TYP.

- The third segment inserts the assembled object module into an object library and runs VDE/DEPEND to collect dependency data. This segment of the DCL command file is the same for all MACRO modules, and the VDE script that generates it has the selection template [*]*.MAR.

One VDE script could generate all three segments of the DCL command file, but then a separate script is needed for each MACRO module in the library. This solution is undesirable because the script keywords that generate the first and third segments of the DCL command file are the same for all MACRO modules, thus wasting space in the VDE database and making it difficult to modify the common parts of the scripts. A better solution is to have two generic scripts with selection templates [*]*.MAR that generate the first and third segments of each MACRO module's DCL command file, and to have a small script specific to each module to generate the second segment. This solution makes it easy to modify the module-independent segments of the keyword files because they are derived from just two VDE scripts.

To allow you to generate different segments of a DCL command file from separate scripts, VDE allows you to assign a *segment number* to each script. A segment number is an integer value in the range from 1 to 20 that is specified with the /SEGMENT qualifier to the CREATE SCRIPT command. When VDE generates a DCL command file from scripts, it looks for an appropriate script for each possible segment number. VDE selects the script for each segment number using the normal script selection rules so that more module-specific or facility-specific scripts are favored over more generic scripts. After selecting the scripts, VDE interprets them in order by segment number to generate the DCL command file. If there are no scripts for a given segment, that segment is simply omitted from the command file. As a result, you can have up to twenty segments in a DCL command file, each generated by a separately selected script.

If you do not specify a segment number for a script, VDE assigns segment number 10 to the script. If your DCL command files do not need multiple segments, you can thus ignore segment numbers and let VDE generate segment 10 (and only that segment) for each command file. If you later decide that you need additional segments before or after the original segment, you have ample segment numbers to choose from in both directions.

Debugging Scripts

The INVOKE SCRIPT command is a useful tool for debugging scripts. INVOKE SCRIPT allows you to fetch and invoke a specified script, and then to examine the DCL command(s) generated by the script.

For further information on this command, see INVOKE SCRIPT.

EXAMPLES

1

```
$ TYPE [MYDIR]SCRIPT_C.VDESCRIPT
! Script to compile a C module.
!
EMIT "$ SET VERIFY"
EMIT "$ DEFINE/NOLOG SRC$ ", $DIR(SRC)
EMIT "$ DEFINE/NOLOG OBJ$ ", $DIR(OBJ)
EMIT "$ CC/DEBUG/OBJ=OBJ$:/ANA=OBJ$: SRC$:",$MODTYP
EMIT "$ VDE/DEPEND C OBJ$:",$MOD,".ANA"
EXIT
$ VDE
VDE> CREATE SCRIPT/COMPILE=[RMS]*.C [MYDIR]SCRIPT_C.VDESCRIPT
%VDE-I-SCRPADDED, script from file DEV$:[MYDIR]SCRIPT_C.VDESCRIPT;9 added to
 the database
 script to compile module [RMS]*.C (segment 10)
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

In this example, the user first types the script file [MYDIR]SCRIPT_C.VDESCRIPT on the terminal. The script contains five EMIT keywords followed by an EXIT keyword. The user then enters VDE and uses the CREATE SCRIPT keyword to enter that script file into the VDE database as the compile script for all C modules in facility RMS. The log messages confirm that the script was successfully entered into the database.

2

```
$ TYPE [MYDIR]SCRIPT_LINK_X.VDESCRIPT
! VDE script to generate the DCL commands needed to link module X.EXE.
!
EMIT "$ SET VERIFY"
EMIT "$ DEFINE OBJLIB ", $DIR(OBJ,CODE)
EMIT "$ LINK/EXE=OBJLIB:X.EXE/MAP=OBJLIB:X.MAP  -"
EMIT "        OBJLIB:OBJLIB.OLB/INCLUDE=C,OBJLIB:OBJLIB.OLB/LIBRARY"
EMIT "$ DEASSIGN OBJLIB"
$ VDE
VDE> CREATE SCRIPT/LINK=X.EXE [MYDIR]SCRIPT_C
%VDE-I-SCRPADDED, script from file DEV$:[MYDIR]SCRIPT_LINK_X.VDESCRIPT;2
 added to the database
 script to link module [CODE]X.EXE (segment 10)
%VDE-I-COMMIT, database transaction has successfully committed
VDE> SHOW SCRIPT/TEXT/LINK=X.EXE
Script to link module [CODE]X.EXE (segment 10)
   Text of script:
     1: ! VDE script to generate the DCL commands needed to link module X.EXE.
     2: !
     3: EMIT "$ SET VERIFY"
     4: EMIT "$ DEFINE OBJLIB ", $DIR(OBJ,CODE)
     5: EMIT "$ LINK/EXE=OBJLIB:X.EXE/MAP=OBJLIB:X.MAP  -"
     6: EMIT "        OBJLIB:OBJLIB.OLB/INCLUDE=C,OBJLIB:OBJLIB.OLB/LIBRARY"
     7: EMIT "$ DEASSIGN OBJLIB"

VDE>
```

This example displays the script file SCRIPT_LINK_X.VDESCRIPT to show the text of the script. The script consists of two comment lines and five EMIT keywords. The EXIT keyword is omitted in this case. The user then enters VDE and uses the CREATE SCRIPT keyword to enter the script into the VDE database. The /LINK qualifier specifies that the script links module X.EXE in the current default facility, facility CODE. By default, the script's segment number is 10. Finally, the SHOW SCRIPT keyword shows that the script can now be displayed directly from the database.

3

```
VDE> CREATE SCRIPT/NEW_STREAM=* SCRIPT_NEW_STRM.VDESCRIPT
%VDE-I-SCRPADDED, script from file DEV$:[MYDIR]SCRIPT_NEW_STRM.VDESCRIPT;3
 added to the database
 script to create new stream for facility * (segment 10)
%VDE-I-COMMIT, database transaction has successfully committed
VDE> SHOW SCRIPT/TEXT/NEW_STREAM=*
Script to create new stream for facility * (segment 10)
   Text of script:
     1: ! VDE script to populate the directories for a facility
     2: ! when a new stream is created.
     3: !
     4: EMIT "$ DEFINE/NOLOG OLDOBJ ", $DIR(OBJ,%CURRENT,%PARENT)
     5: EMIT "$ DEFINE/NOLOG NEWOBJ ", $DIR(OBJ,%CURRENT,%CURRENT)
     6: EMIT "$ COPY OLDOBJ:*.OLB NEWOBJ:"
     7: EMIT "$ COPY ",$DIR(SRC,%CURRENT,%PARENT),"*.H -"
     8: EMIT "       ",$DIR(SRC,%CURRENT,%CURRENT)
     9: EXIT

VDE>
```

This example creates a NEW_STREAM script that will be executed for all facilities when a new development stream is created. The CREATE SCRIPT keyword creates the script and the SHOW SCRIPT keyword displays the script from the database. The script generates DCL keywords to copy all object libraries from the parent stream's object directory to the current stream's object directory. (The ''current'' stream is the new stream being created in this case.) The script also copies all .H files from the parent stream's source directory for the facility to the new stream's source directory for the facility.

4

```
VDE> CREATE SCRIPT/INSERT_GENERATION=[*]*.* [MYDIR]SCRIPT_INS_GEN
%VDE-I-SCRPADDED, script from file DEV$:[MYDIR]SCRIPT_INS_GEN.VDESCRIPT;4 added
 to the database
 script to insert generation for module [*]*.* (segment 10)
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
VDE> SHOW SCRIPT/TEXT/INSERT_GENERATION=[*]*.*
Script to insert generation for module [*]*.* (segment 10)
   Text of script:
     1: ! Script for the INSERT GENERATION command.  This script fetches
     2: ! out a clear copy of the generation just inserted into a stream
     3: ! in order to keep the stream's source directory up-to-date.
     4: !
     5: FETCH $FACMODTYP, "/LOG/OUTPUT=", $DIR(SRC)
     6: EMIT "$ PURGE ", $DIR(SRC),$MODTYP

VDE>
```

This example defines a script to be executed each time the INSERT GENERATION command is performed for any module. The SHOW SCRIPT command displays the text of the script. The script contains a FETCH keyword that fetches the just inserted generation into the SRC subdirectory for the affected stream and facility. This operation maintains an up-to-date copy of the source file in that directory. Because the /LOG qualifier is generated for the FETCH keyword, VDE will print an informational log message as the module is fetched during script execution. The script also contains an EMIT keyword that generates a PURGE keyword to be performed on the file just fetched out.

Script Functions

This section contains the script functions supported by VDE.

This documentation is under construction...

Requires CRESCRP privilege.

DESCRIPTION

Scripts are used to add various user-defined extensions to various VDE functions and commands. VDE invokes various scripts at various times during VDE processing—see Script Types for further information on this.

Scripts are stored in the VDE database, and specific scripts are invoked automatically and transparently at predetermined points. When a script is invoked, the commands within the script—see Script Keywords for further information on the available *script keywords*—are converted into DCL commands and these DCL commands are then executed in a subprocess.

Within a script, *script functions* can be used to extract information from the VDE database and insert that information into the DCL commands generated by by the EMIT script command, or into the command line generated by the FETCH command.

Certain script functions extract names, such as stream, facility, module, and type names from the database, while others extract the OpenVMS directory specifications of various root directories and subdirectories in the VDE library. VDE accepts the following script functions:

- $FAC

  Returns the facility name of the facility currently being processed or of the module currently being processed (for example, the module being compiled).

- $FACMODTYP

  Returns the full module name of the module currently being processed (for example, the module being compiled). This module name includes the facility name, module name, and type name in the normal VDE module-name syntax, for example [FACIL]MOD.TYP.

- $FILEDIR

  Returns the OpenVMS directory specification of the directory where the module currently being replaced is located or where the module currently being fetched or reserved should go. This is either the current default directory or the directory specified with the /INPUT qualifier to the REPLACE command or the /OUTPUT qualifier to the FETCH or RESERVE command. This function is only defined for FETCH, NOKEEP, REPLACE, RESERVE, and STAGING scripts.

- $FILESPEC

Returns the OpenVMS file specification of the input file for the module currently being replaced or of the output file for the module currently being fetched or reserved. This is either the default file specification for the command or the file specification specified with the /INPUT qualifier to the REPLACE command or the /OUTPUT qualifier to the FETCH or RESERVE command. This function is only defined for FETCH, NOKEEP, REPLACE, RESERVE, and STAGING scripts.

- $DIR *(dir-symbol [, fac-name [, stream-name]])*

  Returns the OpenVMS directory specification of a specified facility subdirectory for a given development stream. The **dir-symbol** parameter specifies the VDE directory symbol that identifies the desired subdirectory. Directory symbols are defined with the SET DIRECTORY command. The **fac-name** parameter specifies the facility whose subdirectory should be retrieved. If this parameter is omitted, the facility of the module currently being processed is assumed (for example, the module being compiled). The **stream-name** parameter specifies the development stream for which the subdirectory should be retrieved. If this parameter is omitted, the default stream is assumed.

  You can specify the keyword %CURRENT for the **fac-name** parameter to refer to the facility of the module being processed and you can specify %CURRENT for the **stream-name** parameter to refer to the currently set stream. You can also specify %PARENT for the **stream-name** parameter to refer to the current stream's parent; this keyword is mainly used in NEW_STREAM scripts to refer to the directories of the parent of the new stream being created.

  For additional information on directory symbols (*dir-symbol*), see CANCEL DIRECTORY, SET DIRECTORY, and SHOW DIRECTORY.

- $FACROOT *[(fac-name [, stream-name])]*

  Returns the OpenVMS directory specification of a specified facility root directory for a given stream. The **fac-name** parameter specifies the facility whose root directory should be retrieved. If this parameter is omitted, the facility of the module currently being processed is assumed (for example, the module being compiled). The **stream-name** parameter specifies the development stream for which the given facility root directory should be retrieved. If this parameter is omitted, the default stream is assumed.

  You can specify the keyword %CURRENT for the **fac-name** parameter and you can specify the keyword %CURRENT or %PARENT for the **stream-name** parameter. See the description of the $DIR function for the meanings of these keywords.

- $LIBROOT

  Returns the OpenVMS directory specification of the root directory of the current VDE library. This is the directory that contains the VDE database.

- $MOD

Returns the module name of the module currently being processed (for example, the module being compiled). The module's facility name and type name are not included.

- $MODTYP

  Returns the module name and type name of the module currently being processed (for example, the module being compiled). The two names are separated by a period, as in MOD.TYP. The facility name is not included.

- $SCRATCH

  Returns the device and directory for the current process-specific scratch directory.

- $SESSIONNAME

  Returns the name of the reservation session associated with the current RESERVE, REPLACE, or UNRESERVE command. If no reservation session is used for the command, this function returns the null string. This function is only defined for NOKEEP, REPLACE, RESERVE, STAGING, and UNRESERVE scripts.

- $SESSIONNUM

  Returns the unique session number of the reservation session associated with the current RESERVE, REPLACE, or UNRESERVE command. The session number is returned as a decimal integer value. If no reservation session is used for the command, this function returns the number 0. This function is only defined for NOKEEP, REPLACE, RESERVE, STAGING, and UNRESERVE scripts.

- $STAGEDEST

  Returns the destination directory or file specification for a file being copied to a staging directory while being queued for replacement. If the input file to the REPLACE command has the same file name and type as the VDE module being replaced, the $STAGEDEST function returns the OpenVMS directory specification of the staging directory into which the file should be copied. If the input file has a different file name or type than the VDE module, which can happen if a file name is specified with the /INPUT qualifier, then the $STAGEDEST function returns the staging area directory specification concatenated with the VDE module name and type.

  The $STAGEDEST function is used to specify the destination parameter of the staging script COPY command that copies the input file into the staging area. By leaving the file name off this parameter when the name of the input file is not being changed (the typical case), this function ensures that the COPY command preserves the file's creation date and time. This function is only defined for STAGING scripts.

- $STAGEDIR

Returns the OpenVMS directory specification of the staging directory for the module currently being replaced. Staging directories are used to hold files for queued replacements, and this function is only defined for STAGING scripts.

- $STAGEROOT

  Returns the OpenVMS directory specification of the staging area root directory for the VDE library. (Staging areas hold files for queued replacements.)

- $STREAM

  Returns the name of the default development stream (for example, the stream being created or the stream for which a build job is being run.)

- $STRMDIR *(dir-symbol [, stream-name])*

  Returns the OpenVMS directory specification of a specified stream subdirectory for a given development stream. The **dir-symbol** parameter specifies the VDE directory symbol that identifies the desired subdirectory. Directory symbols are defined with the SET DIRECTORY command. The **stream-name** parameter specifies the stream whose subdirectory should be retrieved. If this parameter is omitted, the default stream is assumed.

  You can specify the keyword %CURRENT or %PARENT for the **stream-name** parameter. See the description of the $DIR function for the meanings of these keywords. For additional information on directory symbols (*dir-symbol*), see CANCEL DIRECTORY, SET DIRECTORY, and SHOW DIRECTORY.

- $STRMROOT *[(stream-name)]*

  Returns the OpenVMS directory specification of a specified stream's root directory. The **stream-name** parameter specifies the development stream whose root directory should be retrieved. If this parameter is omitted, the default stream is assumed.

  You can specify the keyword %CURRENT or %PARENT for the **stream-name** parameter. See the description of the $DIR function for the meanings of these keywords.

- $TYP

  Returns the type name of the module currently being processed (for example, the module being compiled).

For further information on creating scripts, see the CREATE SCRIPT command.

Script Types

This section contains the types of scripts supported by VDE.

Requires CRESCRP privilege.

DESCRIPTION

VDE supports many types of scripts. The different types are distinguished by the commands or circumstances that cause them to be invoked. For example, COMPILE scripts are invoked when you enter the COMPILE command or when you run a build job, while RESERVE scripts are invoked when you enter RESERVE commands.

Some types of scripts are optional because the commands that invoke them will run correctly without scripts. In such cases, the script is there only to allow you to specify optional additional processing for the command. The RESERVE command, for example, does not need RESERVE scripts to work correctly, but if you want the RESERVE command to send mail to selected users whenever someone reserves a module, you must use a RESERVE script to perform that action. Other types of scripts are mandatory. For example, VDE does not know how to compile, copy, or link modules unless you specify how with the appropriate scripts. The COMPILE, COPY, LINK, and BUILD commands thus require such scripts to be defined.

Most scripts are executed in logical isolation from other scripts, and are called in response to a particular command or request. If a particular operation results in multiple propogations or similar operations, certain scripts are invoked once for each propogation.

A few script types, such as the BUILD, COMPILE and LINK scripts, operate as a group. The BUILD script can be used to set up DCL global symbols or logical names that are used as input to subsequent COMPILE or LINK scripts. As partial builds can invoke a subset of all available COMPILE and LINK scripts, DCL global symbols and logical names that are used as inputs into any COMPILE and LINK scripts should only be equated or defined in the BUILD script.

You specify the type of a script with a qualifier. For the CREATE SCRIPT command, you must specify one such qualifier.

Table 3–3 lists and describes the various script types accepted by VDE:

Table 3–3 Script Types

Script Type	Description
BUILD	

Table 3–3 (Cont.) Script Types

Script Type	Description
	The BUILD script sets up optional build-job definitions. Any DCL symbols equated in a BUILD script must be set up as global symbols if they are to remain visible to subsequent COMPILE and LINK scripts. DCL symbols required as inputs to COMPILE and LINK scripts should be defined only in the BUILD script, as any partial builds may not execute a particular subset of the COMPILE or LINK scripts. This script is executed at the start of each build-job process.
COMPILE	
	The COMPILE script contains the command(s) used to compile a module. DCL symbols required as inputs to compile and link scripts should be defined only in the BUILD script, as any partial builds may not execute a particular subset of the COMPILE or LINK scripts. COMPILE scripts are invoked during builds, and by an explicit COMPILE command.
COPY	
	The copy command is used to copy a module. The COPY script is called as part of a build job, and is called as part of the processing of the COPY command.
CREATE_FACILITY	
	The CREATE_FACILITY script contains the command(s) to be executed when a facility is created.
DELETE_GENERATION	
	The DELETE_GENERATION script performs optional processing for the DELETE GENERATION command.
FETCH	
	The FETCH script performs optional processing for the FETCH command.
INSERT_GENERATION	
	The INSERT_GENERATION script performs optional processing for the INSERT GENERATION command.
LINK	
	The link script links a module. DCL symbols required as inputs to compile and link scripts should be defined only in the build script, as any partial builds may not execute a particular subset of the compile or link scripts. This script is called by build jobs and by the LINK command.
NEW_STREAM	
	The new-stream script copies files for a particular facility from the parent stream into a new stream. This script is called when a new stream is created.
NOKEEP	
	The nokeep script performs optional processing for the REPLACE command. Specifically, the nokeep script is called when the REPLACE/NOKEEP command is specified.

Table 3–3 (Cont.) Script Types

Script Type	Description
REPLACE	
	The replace script performs optional processing for the REPLACE or (on queued-replacement streams) the PERFORM REPLACEMENT command.
	If stream-specific reference directories are required, this script can be used to update the reference directory on each replacement.
RESERVE	
	The reserve script performs optional processing when a RESERVE command is entered.
	Should notification of a reservation be required, a site-specific reserve script can be coded to send mail.
SET_FACILITY	
	The set-facility script performs optional processing for the SET FACILITY command.
SET_STREAM	
	The set-stream script performs optional processing for the SET STREAM command.
STAGING	
	The staging script copies a source file to a staging area for the REPLACE command when the replacement is queued.
UNRESERVE	
	The unreserve script performs optional processing for the UNRESERVE command.

For other script-related commands, see CREATE SCRIPT, DELETE SCRIPT, EXTRACT SCRIPT, INVOKE SCRIPT, MODIFY SCRIPT, and SHOW SCRIPT.

SET ARCHITECTURE

Sets...

ARCHITECTURE support is not fully implemented.

FORMAT SET ARCHITECTURE *arch-name*

PARAMETER *arch-name*

The name of the architecture to be set. This name may be up to 39 characters long and must follow VDE name syntax.

DESCRIPTION

Under Construction...

EXAMPLES

1
```
VDE> SET ARCHITECTURE VAXPDP
VDE>
```

Under Construction...

SET CONTEXT

Establishes the context of the current VDE session or the default context for future VDE sessions. A context can be used to specify the default development stream, default architecture and default facility for a VDE session.

FORMAT

SET CONTEXT *ctx-name*

PARAMETER

ctx-name

Specifies the context for the current VDE session or the default context for future VDE sessions. The parameter must be the name of an existing public or private context in the VDE database.

DESCRIPTION

This command establishes a context for the current VDE session or establishes the default context for future sessions. A context sets the following two defaults:

- the default development stream for the VDE session
- the default facility for the session

Using a SET CONTEXT command is equivalent to using individual SET STREAM and SET FACILITY commands to establish the same settings.

If you specify the /DEFAULT qualifier, the context becomes the default for future VDE sessions and VDE records that setting in the database. The context of the current session does not change. If you do not specify the /DEFAULT qualifier, the command establishes the context for the current VDE session.

To display contexts, use the SHOW CONTEXT command.

QUALIFIERS

/DEFAULT

Establishes the specified context as the default context for future VDE sessions. The context of the current session is not changed.

If you do not use this qualifier, the specified context becomes the context for your current session.

/PRIVATE (default)
/PUBLIC

Specifies whether the context to set is a private or public context. The /PRIVATE qualifier specifies that the context is an existing private context. The /PUBLIC qualifier specifies that the context is an existing public context. If neither qualifier is not used, VDE assumes that the specified context is private to you.

Of /PRIVATE and /PUBLIC, /PRIVATE is the default.

EXAMPLES

1

```
VDE> SET CONTEXT BUGFIXES
VDE> SHOW DEFAULTS
Library LIBNAME in root directory DEV$:[PROJDIR]
Default directory is DEV$:[SMITH]
Context BUGFIXES (private to user SMITH)
   Default stream is V2.2-BL1
   Default facility is RMS

VDE>
```

The SET CONTEXT command takes the defaults for the current VDE session from context BUGFIXES. The SHOW DEFAULTS command confirms that those defaults have been established for the session.

2

```
VDE> SET CONTEXT/DEFAULT FOO
VDE>
```

This command makes context FOO the default context for the current user's future VDE sessions.

SET DEVELOPER

Sets or modifies the current settings of the developer flags.

Use this command only under the explicit direction of, and with the assistance of, a VDE developer.

FORMAT

SET DEVELOPER *dev-flags*

DESCRIPTION

Under Construction... Table 3–4 contains various parameters.

Table 3–4 Parameters

Parameter	Description
LOCK	
NOMAIL	
SHOW	
SYNCH	
NONOTES	

EXAMPLES

1
```
VDE> SET DEVELOPER
Under Construction...
VDE> SET DEVELOPER SCRIPT
VDE>
```

SET DIRECTORY

Creates a directory template in the VDE database. Directory templates determine what disk directories VDE creates when creating new streams and facilities and various disk storage areas.

Requires CRESTRM privilege.

FORMAT

SET DIRECTORY *dir-spec*

PARAMETER

dir-spec

Specifies a OpenVMS directory specification for the new directory template. This parameter may specify an absolute or a relative directory specification. An absolute directory specification is a fully specified OpenVMS directory name and follows the normal syntax for such names. You use an absolute directory specification when you want to decide the exact disk location for a directory in the VDE library. For example, you may want to locate a new stream's root directory on a specific disk before VDE creates the stream's directory tree. The parameter DEV:[DIRNAM.FOO] is an example of an absolute directory specification.

A relative directory specification begins with a left bracket and a period ([.), then contains one or more subdirectory names separated by periods, and ends with a right bracket ( ]). This directory specification is concatenated to the name of the appropriate root directory to form a full directory specification. You use a relative directory specification to specify a subdirectory name that VDE should use relative to some root directory. The directory specification [.SRC], for example, can be used to specify that a facility subdirectory named SRC should be created within each facility's root directory.

DESCRIPTION

The SET DIRECTORY command creates a new ''directory template'' in the VDE database. A directory template specifies the name that VDE should assign to each disk directory of a certain kind when creating directories for new streams or facilities. You can create directory templates for stream root directories, facility root directories, facility subdirectories, the library's delta-file root directory, and the library's staging-area root directory for queued replacements. (The library root directory is created when you create the library and cannot be changed thereafter.)

The SET DIRECTORY command does not itself create new directories; it only stores the new directory template in the VDE database. VDE does not use the template to create new disk directories until a subsequent CREATE STREAM or CREATE FACILITY command creates the directory tree for a new stream or facility. The CREATE DIRECTORY_TREE command uses directory templates to create directories for an existing stream or facility.

Unless the new template is for the library's delta files or its staging area, it is only defined for the default development stream. However, directory templates are inherited by the stream's future descendants and are used to create the disk directories for those descendants. They are also used to create directory trees for new facilities in the default stream.

If you want to create directory templates for a new stream, you can specify the templates for the parent stream, in which case they are automatically applied to all child streams. You must then enter the SET DIRECTORY command before you enter the CREATE STREAM command for the new stream. Alternatively, you can create the new stream with the /DEFER qualifier on the CREATE STREAM command. This qualifier suppresses directory-tree creation for the new stream. You can then use the SET STREAM command to make the new stream the default stream and use the SET DIRECTORY command to create the directory templates you want. A subsequent CREATE DIRECTORY_TREE command uses the new templates (and any applicable default rules) to create the directory tree for the new stream. The /DEFER qualifier thus defers directory-tree creation so that you can enter SET DIRECTORY commands before creating the new directory tree.

You may also want to create directory templates for a new facility. If the templates should apply to that facility only, you must create the new facility with the /DEFER qualifier on the CREATE FACILITY command. This qualifier suppresses directory-tree creation for the new facility. You then use the SET DIRECTORY command with the /FACILITY qualifier to create the directory templates you want for that facility. A subsequent CREATE DIRECTORY_TREE command uses the new templates to create the directory tree for the facility.

For additional information on directory creation, and on directory symbols (*dir-symbol*) and directory symbol processing, see CANCEL DIRECTORY, SHOW DIRECTORY, and Script Functions.

QUALIFIERS

/DEFER
/NODEFER (default)

The /DEFER qualifier specifies that directory tree creation for new streams be deferred when those streams are created. This action only applies to new streams for which the default stream is the parent. This qualifier should be used when you always want an opportunity to alter a new stream's directory templates before you actually create its directory tree. The /NODEFER qualifier specifies that directory tree creation not be deferred for new streams unless the user explicitly uses the /DEFER qualifier with the CREATE STREAM command.

/DELTA_FILES
/STAGING
/STREAM (default)

Specifies whether the new directory template is for the VDE library's delta files, for its staging area, or for the default stream. The /DELTA_FILES qualifier specifies that the new directory template is for directories that store the library's delta files (CMS libraries). The /STAGING qualifier

specifies that the new directory template is for the library's staging-area root directory. (Staging areas hold files for queued replacements.) The /STREAM qualifier specifies that the new directory template is for directories that store files for the current default stream. Because streams inherit directory templates from their parents, the /STREAM qualifier causes the new directory template to apply to all of the default stream's future descendants.

Of /DELTA_FILES, /STAGING and /STREAM, /STREAM is the default.

/FACILITY=fac-name

Specifies that the new directory template only applies to the facility given by the **fac-name** parameter. This parameter may not contain wildcard characters. If both this qualifier and the /SUBDIRECTORY qualifier are specified, VDE creates a directory template for the specified facility subdirectory. If the /FACILITY qualifier is specified and the /SUBDIRECTORY qualifier is omitted, VDE creates a template for the specified facility's root directory.

If the /FACILITY qualifier is omitted and the /SUBDIRECTORY qualifier is specified, VDE creates a directory template for the facility subdirectory for all facilities that are not explicitly specified in other templates. If both the /FACILITY qualifier and the /SUBDIRECTORY qualifier are omitted, VDE creates a template for the stream or delta-file root directory.

/LOG (default)
/NOLOG

Controls whether log messages are printed after the directory template is created. The /LOG qualifier causes such messages to be printed and the /NOLOG qualifier suppresses them. These messages indicate that the directory template has been added to the database and that the database transaction has completed successfully.

/PROMPT
/NOPROMPT (default)

The /PROMPT qualifier specifies that users who create new streams for which the default stream is the parent be prompted for the name of the new stream's root directory. This qualifier should be used when you do not want to use the default root directory name for new streams, but want the user to explicitly select the location of the new stream's directory tree each time (perhaps to select a different disk each time). The /NOPROMPT qualifier specifies that a new stream should get the root directory determined by VDE's default rules unless the user explicitly specifies otherwise on the CREATE STREAM command.

/REMARK="string"

Specifies a remark string to be stored for the new directory template. The quoted remark string, which can be up to 132 characters long, may contain any explanatory remark about the new directory template that you want stored in the VDE database.

/SUBDIRECTORY=dir-symbol

Specifies that the new directory template is for a facility subdirectory. The **dir-symbol** parameter specifies the directory symbol that you want to associate with the subdirectory. The directory symbol must follow normal VDE name syntax. You use the directory symbol in VDE scripts to reference the subdirectory; see the description of the CREATE SCRIPT

command for an explanation of VDE scripts. If the /SUBDIRECTORY qualifier is omitted, VDE creates a directory template for a facility, stream, or delta-file root directory. (The description of the /FACILITY qualifier explains how the qualifiers interact to select the template to create.)

When you specify facility subdirectories for delta-files, the directory symbol may have a special meaning to VDE. If you specify a delta-file subdirectory with the symbol VDE$CMS, VDE puts the CMS library for the facility in that subdirectory instead of in the facility root directory. If you specify a delta-file subdirectory with the symbol VDE$MARKER, VDE puts the actual source files for modules that use ''marker files'' in that subdirectory. If you use marker files, you must have both a marker-file subdirectory and a CMS-library subdirectory. (The description of the /MARKER qualifier to the CREATE MODULE command discusses marker files in more detail.)

EXAMPLES

1

```
VDE> SET DIRECTORY/STREAM/SUBDIRECTORY=ANASYM [.ANA]
%VDE-I-DIRADDED, directory specification [.ANA] added to the database
%VDE-I-COMMIT, database transaction has successfully committed
VDE> SHOW DIRECTORY
Directory-creation templates for stream V2.1-1:
   Directory creation flags:
      Do not defer creation of directory tree for substreams
      Do not prompt for root directory for substreams

   Stream root directory is [.V2_1-1] by default
      Subdirectory:   [.VDE$BLD] (symbol VDE$BLD)

   Directory templates for unspecified facilities:
      Subdirectory:   [.ANA] (symbol ANASYM)
      Subdirectory:   [.OBJ] (symbol OBJ)
      Subdirectory:   [.SRC] (symbol SRC)
      Subdirectory:   [.VDE$COM] (symbol VDE$COM)
      Subdirectory:   [.VDE$LOG] (symbol VDE$LOG)

   All other directories created using default rules

VDE>
```

This example creates a directory template for a new facility subdirectory. The directory specification is [.ANA], which says that a subdirectory called ANA should be created within each facility's root directory. The directory symbol associated with that subdirectory is ANASYM. This symbol is used in VDE scripts to reference the subdirectory. The SHOW DIRECTORY command shows that the new directory template is stored in the database.

```
2  VDE> SET DIRECTORY/FACILITY=RMS [MYDIR.RMS]
   %VDE-I-DIRADDED, directory specification [MYDIR.RMS] added to the database
   %VDE-I-COMMIT, database transaction has successfully committed
   VDE> SHOW DIRECTORY
   Directory-creation templates for stream V2.1-1:
      Directory creation flags:
         Do not defer creation of directory tree for substreams
         Do not prompt for root directory for substreams

      Stream root directory is [.V2_1-1] by default
         Subdirectory:   [.VDE$BLD] (symbol VDE$BLD)

      Directory templates for unspecified facilities:
         Subdirectory:   [.OBJ] (symbol OBJ)
         Subdirectory:   [.SRC] (symbol SRC)
         Subdirectory:   [.VDE$COM] (symbol VDE$COM)
         Subdirectory:   [.VDE$LOG] (symbol VDE$LOG)

      Directory templates for facility RMS:
         Root directory: [MYDIR.RMS]

      All other directories created using default rules

   VDE>
```

This example creates a directory template for the root directory of facility RMS. The directory specification is [MYDIR.RMS], an absolute directory specification for the root directory. VDE will create this directory for facility RMS when a subsequent CREATE DIRECTORY_TREE command is entered for that facility or a new stream is created from the default stream. The SHOW DIRECTORY command shows that the new directory template is stored in the database.

The following example shows an example of a VDE library creation, including a number of directory-related commands.

```
$
$! Locate and delete previous incarnations of the test facility files,
$! then unpack the saveset containing the test files...
$
$ if f$search("sys$scratch:A.REQ") .nes. "" Then Delete sys$scratch:A.REQ;*/lo
$ if f$search("sys$scratch:B.REQ") .nes. "" Then Delete sys$scratch:B.REQ;*/lo
$ if f$search("sys$scratch:C.B32") .nes. "" Then Delete sys$scratch:C.B32;*/lo
$ if f$search("sys$scratch:D.B32") .nes. "" Then Delete sys$scratch:D.B32;*/lo
$ if f$search("sys$scratch:E.B32") .nes. "" Then Delete sys$scratch:E.B32;*/lo
$ if f$search("sys$scratch:F.B32") .nes. "" Then Delete sys$scratch:F.B32;*/lo
$
$ Backup VDE$CREATE.BCK/SAVE Sys$Scratch:
$
$! Create the core directory -- this must be present...
$
$ Create/Directory STAR$DATA:[QTV.WWW.VDE$DATABASE]
$ VDE
!
! create the WWW library in STAR$DATA:[QTV.WWW...], using the
! mnemonic WWW, placing all Rdb-related library components in
! STAR$DATA:[QTV.WWW.VDE$DATABASE].
!
create librar  STAR$DATA:[QTV.WWW.VDE$DATABASE] /DEFER -
 /NAME=WWW -
 /REMARK="QTV WWW Source Library" -
 /NOAUTO_CONVERT /AUTO_ADD_USER=VMS_SOURCE /AUTO_MAIL -
 /ASK_INFO_FILE /ASK_REPL_CAUSE /SHOW_GEN_EXPR /NOSTATISTI -
 /NOHISTORY_NOTES /NOALLOW_DELETE /NOWILD_INS_GEN -
 /NOCMS_ELEM_HIST /CREATOR="Stephen Hoffman" -
 /DEF_PRIVILEGES=(CREMOD,RESREP,CREFAC) -
 /AUTH_PRIVILEGES=(CREMOD,RESREP,CREFAC) -
 /data=STAR$DATA:[QTV.WWW.VDE$DATABASE] -
```

```
 /journ=STAR$DATA:[QTV.WWW.VDE$DATABASE] -
 /snap=STAR$DATA:[QTV.WWW.VDE$DATABASE]
!
SHOW DEFAULTS
SHOW LIBRARY/FULL
MODIFY CONTEXT/PUBLIC DEFAULT/NOSTREAM/NOFACILITY
!
CREATE USER SYSTEM /NAME="System Manager" -
    /DEF_P=ALL/AUTH_P=ALL/REMARK="System Manager"
!
! Create the CMS ("delta") library directory structure.
! Configure the CMS libraries to use the directory structure:
!   STAR$DATA:[QTV.WWW.CMS$DATABASE.'facility'.CMS].
! Configure the "marker" files to use the directory structure:
!   STAR$DATA:[QTV.WWW.CMS$DATABASE.'facility'.SRC]...
!
SET DIRECTORY/DELTA_FILES STAR$DATA:[QTV.WWW.CMS$DATABASE]
SET DIRECTORY/DELTA_FILES/SUBDIRECTORY=VDE$CMS [.CMS]
SET DIRECTORY/DELTA_FILES/SUBDIRECTORY=VDE$MARKER [.SRC]
CREATE DIRECTORY_TREE/DELTA_FILES
SHOW DIRECTORY/DELTA_FILES
!
! Create the staging area...
!
SET DIRECTORY/STAGING STAR$DATA:[QTV.WWW.VDE$STAGING]
CREATE DIRECTORY_TREE/STAGING
!
! Create the stream/build directories
!
SET DIRECTORY/STAGING STAR$DATA:[QTV.WWW.VDE$STREAM]
CREATE DIRECTORY_TREE/STREAM
!
! create and load the VDE test facility.
!
CREATE FACILITY AAA_VDE_TEST -
    /REMARK="Test facility for VDE developers"
SET FACILITY AAA_VDE_TEST"
CREATE MODULE/NODELETE/REMARK="VDE self-test module" -
  /INPUT=SYS$SCRATCH: A.REQ
CREATE MODULE/NODELETE/REMARK="VDE self-test module" -
  /INPUT=SYS$SCRATCH: B.REQ
CREATE MODULE/NODELETE/REMARK="VDE self-test module" -
  /INPUT=SYS$SCRATCH: C.B32
CREATE MODULE/NODELETE/REMARK="VDE self-test module" -
  /INPUT=SYS$SCRATCH: D.B32
CREATE MODULE/NODELETE/REMARK="VDE self-test module" -
  /INPUT=SYS$SCRATCH: E.B32
CREATE MODULE/NODELETE/REMARK="VDE self-test module" -
  /INPUT=SYS$SCRATCH: F.B32
!
EXIT
$
$ Exit
```

SET EDITOR

Specifies the editor that VDE should use for those VDE commands that invoke an editor.

FORMAT

SET EDITOR *"editor-string"*

PARAMETER

editor-string

The DCL command that VDE should spawn to invoke the desired editor. This string should include the editor name and any desired qualifiers. VDE will append the name of the file to edit to this string when invoking the editor.

DESCRIPTION

The SET EDITOR command specifies the editor that VDE should use when a subsequent VDE command must invoke an editor for you. VDE can invoke an editor for you when you review queued replacements, for example, or when you create a comment file with the ACCEPT or REJECT REPLACEMENT command. In such cases, VDE spawns a DCL command to place you in an editing session. The spawned DCL command is formed by concatenating your editor string, a space, and the name of the file to edit. The editor string that you specify with the SET EDITOR command should thus give the DCL command that invokes your preferred editor, along with any qualifiers that you want for that command.

VDE maintains two editor strings in memory for the current VDE session. One string is used for read-write editing and the other for read-only editing. Use the /READ_ONLY qualifier to specify the read-only editor string. If you do not specify a read-only editor string (or if you specify the null string for this case), VDE assumes that the editor you specified with the read-write editor string will accept the /READ_ONLY qualifier (most Digital editors do). In this case, VDE appends the /READ_ONLY qualifier to your read-write editor string to form the read-only editor string. For example, if you specify TPU as the read-write editor string but no read-only editor, then VDE uses the string TPU/READ_ONLY for read-only editing.

The SET EDITOR command always modifies the read-write or read-only editor string for the current VDE session. If you specify the /DEFAULT qualifier, the command in addition writes your current editor settings to the VDE database to establish your default editor strings for future VDE sessions. VDE retrieves these editor strings each time you connect to the VDE database.

If you do not use the SET EDITOR command to declare your preferred editor, VDE uses the system default text editor.

QUALIFIERS

/DEFAULT

Specifies that VDE should write the editor strings for your current session to the VDE database and that those strings should be your default editor strings for future VDE sessions.

/LOG (default)
/NOLOG

Controls whether log messages are printed after your editor strings are written to the VDE database. The /LOG qualifier causes such messages to be printed and the /NOLOG qualifier suppresses them. These messages, which only appear if you specify the /DEFAULT qualifier, indicate that the editor strings have been updated in the database and that the database transaction has successfully committed.

/READ_ONLY

Specifies that you are establishing the editor string to use for read-only editing. If this qualifier is omitted, VDE establishes the editor string to use for read-write editing.

EXAMPLES

1

```
VDE> SET EDITOR "LSEDIT"
VDE> SHOW EDITOR
Current editor strings:
   Read-write editor string: LSEDIT
   Read-only editor string:  LSEDIT/READ_ONLY
VDE>
```

This example establishes LSEDIT (which invokes the Language-Sensitive Editor) as your read-write editor string. The SHOW EDITOR command confirms that this is now your current read-write editor string, and it shows that VDE will append the /READ_ONLY qualifier to this string for read-only editing.

2

```
VDE> SET EDITOR/READ_ONLY "EDIT/EDT/READ"
VDE> SHOW EDITOR
Current editor strings:
   Read-write editor string: LSEDIT
   Read-only editor string:  EDIT/EDT/READ
VDE>
```

This example defines the editor string to use for read-only editing.

3

```
VDE> SET EDITOR/DEFAULT "TPU"
%VDE-I-EDITUPDATE, default editor definition updated in the VDE database
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
VDE> SHOW EDITOR/FULL
Current editor strings:
   Read-write editor string: TPU
   Read-only editor string:  TPU/READ_ONLY
Default editor strings:
   Read-write editor string: TPU
   Read-only editor string:  TPU/READ_ONLY
VDE>
```

This example establishes TPU as your editor. Because the /DEFAULT

qualifier is specified, VDE then writes your current editor strings to the VDE database to be used in future VDE sessions. The SHOW EDITOR command shows that these settings have been established both for the current session and as the default for future sessions.

SET FACILITY

Establishes the default facility for the current VDE session. The default facility is the facility to which a module is assumed to belong when no facility is specified in the module name. The default facility also can be assigned with the SET CONTEXT command.

FORMAT

SET FACILITY *facil-name*

PARAMETER

facil-name

Specifies the name of the facility that becomes the default facility for the current VDE session.

QUALIFIERS

None.

EXAMPLES

1

```
VDE> SET FACILITY RMS
VDE> SHOW MODULE MOD1.MAR
Module [RMS]MOD1.MAR
VDE> SHOW DEFAULTS
Library LIBNAME in root directory DEV$:[PROJDIR]
Default directory is DEV$:[SMITH]
Current session context:
   Default stream is MAIN
   Default facility is RMS
VDE>
```

The SET FACILITY command makes facility RMS the default facility for the current VDE session. The SHOW MODULE command then specifies a module name without a facility name. The module is thus assumed to belong to facility RMS. The SHOW DEFAULTS command also shows that RMS has been established as the default facility in the current session.

SET KEY

Sets and locks the key-definition state for keys defined with the DEFINE/KEY command.

FORMAT

SET KEY

DESCRIPTION

Using the DEFINE/KEY command you can create as many definitions for a key as you want. However, each definition of the key must be assigned a separate context or **state**. For example, if you create five definitions for the PF4 key you must create five different states, one for each definition of the key. If you do not specify its state before you process a definition, the definition does not take effect.

The SET KEY command lets you switch from the current state to any other state; you then can use the key definitions created for that state.

QUALIFIERS

/LOG (default)
/NOLOG

Controls whether the system displays a message indicating that the key state has been set. /LOG causes such a message to be displayed and /NOLOG suppresses it.

/STATE=state-name
/NOSTATE (default)

Specifies the name of the state to be set. The state name can be any alphanumeric string. If you omit the /STATE qualifier or use /NOSTATE, the current state is left unchanged.

EXAMPLES

1
```
VDE> SET KEY/STATE=GOLD
```

The SET KEY command in this example sets the key state to GOLD. You can now use the key definitions that are defined for the GOLD state.

SET LIBRARY

Connects the current VDE session to a specified VDE library.

FORMAT

SET LIBRARY *dir-spec*

PARAMETER

dir-spec

The name of the root directory of a VDE library. The root directory is the disk directory that contains the library's database. VDE connects the current VDE session to the specified library. Use the OpenVMS directory specification format to specify this parameter.

VDE also accepts mnemonic names for the libraries, based on the contents of various logical names, and the list of libraries and the particular mnemonics can be tailored for individual users, for groups of users, and can be defaulted on a system-wide basis.

DESCRIPTION

The SET LIBRARY command disconnects your VDE session from the current VDE library and connects it to the VDE library whose root directory or mnemonic you specify. Subsequent VDE commands then act on the specified library.

To connect to the specified library, the SET LIBRARY command first defines logical name VDE$LIBRARY to point to the specified directory. It then connects to the library you specified and reads its database.

Based on the information contained in the VDE library database, VDE checks to see if you are authorized to use the new library. If you are presently listed as an authorized user of the library, if the library is set to allow the automatic addition of users, or if you should possess the OpenVMS BYPASS privilege, VDE permits access, and you can continue your VDE session normally.

Use of the OpenVMS privilege override is audited, and an informational message will be displayed.

VSC also supports the SET LIBRARY command, but it defaults to the OpenVMS VAX library.

Automatic Library Format Conversion: This command can cause an automatic library conversion if the format of the VDE library predates the version of VDE currently in use. Automatic library conversion will only be performed if the library has the automatic conversion option enabled. See the SHOW LIBRARY, CONVERT LIBRARY and the MODIFY LIBRARY/[NO]AUTO_CONVERT commands.

Should the filename, file extension and/or file version number be erroneously specified in the parameter, the SET LIBRARY command will silently ignore them. Only the device and directory specifications are germane to the command; the appropriate filename and file extension will be appended to the device and directory specification.

For further information on mnemonics, see the VDE$LIBRARY_ DEFAULTS, VDE$LIBRARY_DEFAULTS_LABELS, and VDE$LIBRARY_ DEFAULTS_MNEMONICS logical names. To see the list of mnemonics available, and the translations associated with each, use the following DCL commands:

If the VDE$LIBRARY_DEFAULTS_MNEMONICS logical name is defined, SET LIBRARY commands are available via keyboard F17, F18, F19 and F20 keys. Up to twelve keyboard definitions can be present, depending on the number of translations of VDE$LIBRARY_DEFAULTS_MNEMONICS present; the first four mnemonics are available in the *default* keypad state, the second four in the *gold* state, and the third four in the *blue* state.

```
$
$! These logical names can be (re)defined in the site-
$! specific VDE system startup command procedure, or on
$! a process-, job-, group- or system-specific basis.
$
$! Each logical name is a search list, and the first
$! entry in VDE$LIBRARY_DEFAULTS corresponds to the
$! first entry in VDE$LIBRARY_DEFAULTS_MNEMONICS, and
$! in VDE$LIBRARY_DEFAULTS_LABELS, etc.
$!
$! Display the list of known/predefined libraries:
$ SHOW LOGICAL VDE$LIBRARY_DEFAULTS
$
$! Display the mnemonic name of each library
$ SHOW LOGICAL VDE$LIBRARY_DEFAULTS_MNEMONICS
$
$! Display the textual name of each library
$ SHOW LOGICAL VDE$LIBRARY_DEFAULTS_LABELS
$
```

QUALIFIERS

None.

EXAMPLES

1

```
VDE> SET LIBRARY DEV$:[PROJDIR]
VDE>
```

This example connects the VDE session to the VDE library whose root directory is DEV$:[PROJDIR]. All subsequent VDE commands act on that library until the current library is again changed.

SET LOG

Records VDE input and output in a log file.

FORMAT

SET LOG *[file-spec]*

PARAMETER

file-spec

The name of the log file to be created. All subsequent VDE input and output is recorded in this file until logging is disabled. **File-spec** is any valid OpenVMS file specification. The default file extension is LOG. If the file specification is omitted, the default file name is VDELOGFILE.LOG.

DESCRIPTION

The SET LOG command records all VDE input and output in the specified log file. Input is recorded as entered. Output is recorded with a leading exclamation point (!). Because the exclamation point is the VDE comment character, you can use the log file as a command procedure to rerun the logged VDE session. Use the CANCEL LOG command to disable input and output logging.

QUALIFIERS

None.

EXAMPLES

1

```
VDE> SET LOG MYLOG
VDE> SHOW LOG
Output logging on file: DEV$:[MYDIR]MYLOG.LOG;1
VDE>
```

The SET LOG command creates the VDE log file [MYDIR]MYLOG.LOG. All subsequent commands and their outputs are recorded in this file. The SHOW LOG command confirms that output is being logged to this file.

SET NOVERIFY

Disables the display (or echo) of command input to the terminal.

FORMAT SET NOVERIFY

PARAMETERS

None.

DESCRIPTION

The SET NOVERIFY command disables the display of command input to the terminal. It cancels the effect of the SET VERIFY command. The SET NOVERIFY and CANCEL VERIFY commands are equivalent.

QUALIFIERS

None.

EXAMPLES

1

```
VDE> SET VERIFY
VDE> SHOW MODULE MOD1.FOR
SHOW MODULE MOD1.FOR
Module [FACIL]MOD1.FOR
VDE> SET NOVERIFY
SET NOVERIFY
VDE> SHOW FACILITY FACIL
Facility FACIL
VDE>
```

The SET VERIFY command enables command echoing. Therefore, the SHOW MODULE command is displayed on the terminal before the output of the command is displayed. The SET NOVERIFY command is also echoed but the command disables the display of subsequent commands. The SHOW FACILITY command that follows is not echoed on the terminal; only its output is displayed.

SET PRIVILEGES

Enables VDE privileges for the current VDE session.

FORMAT

SET PRIVILEGES *priv [, priv...]*

PARAMETER

priv

The name of a privilege enabled for the current VDE session. You can enable only those privileges for which you have authorization. To list the privileges for which you are authorized, use the SHOW PRIVILEGES/FULL command. Table 3–5 contains a list of VDE privileges.

Table 3–5 Privileges

Privilege	Description
BUILD	Permits a user to create build jobs in this library.
CRECTX	Allows a user to create, modify, and delete contexts.
CREFAC	Allows the creation and modification of facilities.
CREGRP	Permits the creation, modification, and the deletion of groups.
CREMOD	Allows the creation and modification of modules.
CRESCRP	Required for the creation, modification, and deletion of scripts.
CRESTEP	Allows the creation, modification, and the deletion of build steps.
CRESTRM	Permits the creation and modification of streams.
CREUSR	Permits the creation, modification, and the deletion of users.
DELFAC	Allows the deletion of facilities.
DELMOD	Allows the deletion of modules.
DELREP	Permits the deletion of replacements.
DELSTRM	Permits streams to be deleted.
FROZEN	Allows the user to peplace modules into streams marked frozen.
MODLIB	Allows the user to modify attributes associated with this library.
MODSTRM	Allows the modification of streams.
PERFREP	Allows the user to dequeue queued replacements.
RESREP	Allows the user to reserve and replace modules, and to manipulate fold records.
UNCLOSE	Permits the user to open a stream currently marked as closed.
USERNAME	Allows the user to reserve and replace for other users.
Aggregate Privilege Keywords:	
ADMIN	The privileges usually associated with a project administrator privileges; all of the above-listed privileges except DELFAC, DELMOD, DELREP, DELSTRM, FROZEN, and UNCLOSE.

Table 3–5 (Cont.) Privileges

Privilege	Description
ALL	All privileges.
NONE	No privileges.

The project administrator privileges are those that a project administrator needs to add users to the library; to create facilities, modules, and streams; to run build jobs; and to administer the project.

DESCRIPTION

The VDE privileges enabled for the current session determine the VDE commands (and, in some cases, command qualifiers) you are allowed to use. For example, you must have the CREFAC privilege to use the CREATE FACILITY and MODIFY FACILITY commands. Likewise, you must have the USERNAME privilege to replace modules for another user using the /USERNAME qualifier of the REPLACE command.

When you start a VDE session, the default privileges listed in your user table record in the database become the initial privileges for your session. After that, you can set additional privileges for the current session using the SET PRIVILEGES command if you are authorized to set those privileges. You can also cancel privileges for the current session using the CANCEL PRIVILEGES command and you can display your current privileges using the SHOW PRIVILEGES command.

Users holding the OpenVMS privilege BYPASS are allowed to /OVERRIDE the VDE privileges.

Use of a privilege override is audited, and an informational message will be displayed.

To alter the default or authorized privileges associated with a particular username in the current library, use the MODIFY USER username/DEF_ PRIVILEGES or MODIFY USER username/AUTH_PRIVILEGES command.

QUALIFIERS

/ALL

Enables all authorized VDE privileges. To alter the library-specific authorized privileges for a user, use the MODIFY USER username/AUTH_ PRIVILEGES command.

/DEFAULT

Resets the VDE privilege mask to the default setting for the current user—the default settings can be displayed using the SHOW USER username/FULL command. To alter the library-specific default privileges for a user, use the MODIFY USER username /DEF_PRIVILEGES command.

/OVERRIDE

If the user has the OpenVMS privilege BYPASS enabled, this qualifer overrides all VDE privilege restrictions and enables all VDE privileges.

Use of a privilege override is audited, and an informational message will be displayed.

EXAMPLES

1

```
VDE> SET PRIVILEGES CREFAC, CREMOD
VDE> SHOW PRIVILEGES
Currently set privileges:
      BUILD CREFAC CREGRP CREMOD CRESTEP RESREP
VDE> SET PRIVILEGES/ALL
VDE>
```

The SET PRIVILEGES command sets the CREFAC and CREMOD privileges. The CREFAC privilege allows the user to create and modify facilities in the current library and the CREMOD privilege allows the user to create and modify modules in the library. The SHOW PRIVILEGES command (see SHOW PRIVILEGES) confirms that the privileges have been added to the current privilege mask.

SET PROMPT

Changes the VDE prompt string from VDE> to a string of your choice.

FORMAT

SET PROMPT *prompt-string*

PARAMETER

prompt-string

Specifies the string that is to become the new prompt. If the string contains blanks, special characters, or lowercase letters, you must enclose it in quotation marks (**"**).

DESCRIPTION

The SET PROMPT command lets you to change the VDE prompt to one of your choosing.

EXAMPLES

1

```
VDE> SET PROMPT "Hello Dolly: "
Hello Dolly:
```

In this example, the VDE prompt is changed to ‘‘Hello Dolly:’’. VDE uses this string to prompt for the next command.

SET STREAM

Establishes the default development stream for the current VDE session. Specifying a stream determines the group definitions and module-dependency data VDE uses for the current session: group definitions and module-dependency data are stream-specific. Establishing the default stream also determines what stream commands such as CREATE GROUP, SHOW GROUP, COMPILE, and BUILD operate on; those commands execute only in the default stream.

The default stream also can be specified with the SET CONTEXT command.

FORMAT

SET STREAM *stream-name*

PARAMETER

stream-name
Specifies the name of the default stream for the VDE session.

QUALIFIERS

None.

EXAMPLES

1

```
VDE> SET STREAM V5.3-FT2
VDE> SHOW DEFAULTS
Library LIBNAME in root directory DEV$:[PROJDIR]
Default directory is DEV$:[SMITH]
Current session context:
   Default stream is V5.3-FT2
   Default facility is RMS
VDE>
```

The SET STREAM command establishes stream V5.3-FT2 as the current stream for the VDE session. The SHOW DEFAULTS command shows that V5.3-FT2 has been set as the default stream for the session.

SET TERMINAL

Changes the line width and page size that VDE uses to format output.

FORMAT SET TERMINAL

PARAMETERS

None.

DESCRIPTION

The SET TERMINAL command sets the number of characters per line and the number of lines per page that VDE uses to format output. It does not change the settings of your terminal. The default line width and page size are determined by your device type or by the last SET TERMINAL command you entered at the DCL level.

QUALIFIERS

/PAGE=n

Specifies the number of lines on a page. Normally *n* is 24 or the page size of your workstation, but you can use any value from 11 to 100.

/WIDTH=n

Specifies the maximum number of characters on a line. Typically *n* is 80 or 132 but you can use any value from 20 to 255.

EXAMPLES

```
1  VDE> SET TERMINAL/WIDTH=132
```

This command specifies that the length of a line is 132 characters.

SET VERIFY

Enables the display (or echo) of command input to the terminal.

FORMAT SET VERIFY

PARAMETERS

None.

DESCRIPTION

The SET VERIFY command enables the display of command input to the terminal. This display is useful when you want to see the commands issued by a command procedure before they are executed. You can disable the display with the CANCEL VERIFY or SET NOVERIFY command.

QUALIFIERS

None.

EXAMPLES

1

```
VDE> SET VERIFY
VDE> @X.VDE
@X.VDE
SET FACILITY COPY
SHOW MODULE FOO
Module [FACIL]FOO.MAR
VDE>
```

The SET VERIFY command enables command echoing. Therefore, the next command (@X.VDE) is displayed on the terminal. Then a SET FACILITY and a SHOW MODULE command are read from the command procedure X.VDE. Both commands are displayed on the terminal. The output of the SHOW MODULE command follows the echoed command line.

SHOW ARCHITECTURE

Shows one or more architecture table entries in the VDE database. Each architecture table entry...

ARCHITECTURE support is not fully implemented.

FORMAT

SHOW ARCHITECTURE *arch-name [, arch-name...]*

PARAMETER

arch-name

The name of the architecture to be displayed. This name may be up to 39 characters long and must follow VDE name syntax.

DESCRIPTION

Under Construction...

QUALIFIERS

/BRIEF

Under Construction...

/CURRENT

Under Construction...

/FULL

Under Construction...

/OUTPUT=

Under Construction...

/TOTAL

Under Construction...

EXAMPLES

1

```
VDE> SHOW ARCHITECTURE VAXPDP
VDE>
```

Under Construction...

SHOW BUILD_JOB

Displays information about one or more specified build jobs.

FORMAT

SHOW BUILD_JOB *[stream-name [, stream-name...]]*

PARAMETER

stream-name

The name of a development stream. Information about one or more build jobs in this stream is displayed. Use the /IDENTIFICATION qualifier to specify the build jobs to be displayed. If you do not specify this qualifier, information about the most recent build job for the stream is displayed.

You can display build jobs for more than one stream by using wildcard characters in the stream name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Build jobs are displayed for those streams whose names match the wildcard pattern.

If you omit the **stream-name** parameter, VDE displays information about one or more build jobs for the default stream.

DESCRIPTION

The SHOW BUILD_JOB command displays information about specified build jobs. You can specify build jobs with the **stream-name** parameter and the /IDENTIFICATION qualifier. By default, the SHOW BUILD_JOB command displays information about the most recent build job belonging to the default stream.

The SHOW BUILD_JOB command displays the following information for each build job:

- its identification number
- its stream name
- its status

Optionally, the command can also show all other information available in the VDE database about the build job, including:

- the number of build steps in each state (the number of steps that are not yet started, are running, have completed with errors, and have completed successfully) and the total number of build steps in the job
- the type of build, full or minimal
- the remark string
- the name of the person who started the build job
- the number of build steps in the build job

- the dates and times of transitions for the build job (for example, when the job was entered on the execution queue, when it started, and when it stopped)

The command can also display the list of batch queues and process counts used to run the build job, and it can show all build steps that make up the build job.

QUALIFIERS

/BEFORE=date-time

Displays only those build jobs created before the specified date and time. The date and time can be stated using standard OpenVMS date-time format or can be one of the following keywords: YESTERDAY, TODAY, or TOMORROW. If you use a space to separate the date from the time, remember to enclose the entire date-time string in double quotes. For further information about specifying OpenVMS date-time format, see the OpenVMS DCL Concepts.

/BRIEF (default)
/SUMMARY
/FULL

Determines the type of information displayed about each build job.

The /BRIEF qualifier displays the build-job identification number, the stream name, and status of each build job.

The /SUMMARY qualifier displays the information provided by the /BRIEF qualifier, but it also displays the number of build steps currently in each state (that is, the number of steps that are not yet started, are running, have completed with errors, and have completed successfully) so that you can follow the progress of the build job. The /SUMMARY qualifier also prints the total number of steps in the build job.

The /FULL qualifier displays all information in the database about the build job except queue information and build-step information. (To see queue information, use the /QUEUE qualifier; to see step information, use the /STEP qualifier.)

See the Example section of this command for sample output of the /BRIEF, /SUMMARY, and /FULL qualifiers. Of /BRIEF, /SUMMARY and /FULL, /BRIEF is the default.

/IDENTIFICATION=id-range

Specifies the build-job identification numbers of the build jobs to be displayed for the specified stream. The **id-range** parameter can be a single integer value, a range of integers, or a wildcard character. If the parameter is a single integer (for example, /IDENT=5), VDE displays the build job with that identification number. If the parameter is a range of integers (specified as two build-job identification numbers separated by a colon as in /IDENT=3:5), VDE displays all build jobs with identification numbers in that range. Finally, if the **id-range** parameter is an asterisk (as in /IDENT=*), VDE displays all build jobs for the specified stream.

If you specify a zero or negative build-job identification number in the **id-range** parameter, that number is relative to the identification number of the most recent build job for the stream. For example, if the most recent build job is job 5, then /IDENT=0 is equivalent to /IDENT=5 and /IDENT=–1 is equivalent to /IDENT=4.

If you do not use the /IDENTIFICATION qualifier, VDE displays the most recent build job for the specified stream.

/QUEUE

Displays queue information for the specified build jobs. This information includes the name of each batch queue used to run the build job, the process count for each queue, and the total process count for the build job.

/SINCE=date-time

Displays only those build jobs created after the specified date and time. The date and time can be stated using standard OpenVMS date-time format or can be one of the following keywords: YESTERDAY, TODAY, or TOMORROW. If you use a space to separate the date from the time, remember to enclose the entire date-time string in double quotes. For further information about specifying OpenVMS date-time format, see the OpenVMS DCL Concepts.

/STATUS=(status [, status...])

Displays only those build jobs that have one of the specified status values. Each **status** parameter must be one of the following keywords: NOTQUEUED, QUEUED, RUNNING, SUCCESS, ERRORS, STOPPED, SUSPENDED, and TIMEOUT.

If you specify only one **status** parameter, the parentheses may be omitted.

/STEP [=(status [, status...])]

Displays build-step information for each specified build job. For each build step in the build job, this qualifier displays the type of the build step (such as compile, copy, or link), the name of the module that the step is applied to, and the status of the step. If the /FULL qualifier is also specified, the /STEP qualifier additionally displays the wait count of each step (the number of steps that must complete before this step can execute) and the names of the steps that must wait for the current step to complete.

If you specify one or more **status** parameters, the /STEP qualifier displays only those build steps that have one of the specified status values. Each **status** parameter must be one of the following keywords: NOTSTARTED, RUNNING, SUCCESS, or ERRORS.

If you specify only one **status** parameter, the parentheses may be omitted. If you omit the **status** parameters, all build steps in the specified build jobs are displayed.

/TOTAL

Shows the total number of build jobs that match the specified constraints. These constraints are the stream name, the range of build job identification numbers specified by the /IDENTIFICATION qualifier, the build job status values specified by the /STATUS qualifier, and the time constraints specified by the /BEFORE and /SINCE qualifiers. No information about the individual build jobs is shown; only the total count is displayed. A separate count is provided for each **stream-name** parameter.

EXAMPLES

1

```
VDE> SHOW BUILD_JOB
Build job 11 for stream MAIN has completed with errors
VDE>
```

This example shows the brief form of the SHOW BUILD_JOB output. The build job identification number (11) and stream name (BL22) of the build job are displayed, along with the status of the job (suspended). Because no stream name or /IDENTIFICATION qualifier is specified with the command, the most recent build job for the default stream is displayed.

2

```
VDE> SHOW BUILD_JOB/SUMMARY
Build job 11 for stream MAIN has completed with errors
   Number of build steps not yet started:             0
   Number of currently running build steps:           0
   Number of build steps completed with errors:       1
   Number of successfully completed build steps:      4
   Total number of build steps in build job:          5
VDE>
```

This example shows the summary output of the SHOW BUILD_JOB command. In addition to identifying the build job, the output shows how many build steps are in each state and gives the total number of build steps in the job. In this example, one step completed with errors and all other steps completed successfully.

3

```
VDE> SHOW BUILD_JOB/FULL
Build job 11 for stream BL22 has completed with errors
   Number of build steps not yet started:             0
   Number of currently running build steps:           0
   Number of build steps completed with errors:       1
   Number of successfully completed build steps:      4
   Total number of build steps in build job:          5
   Remark: Build base level 22 once again
   Created by user JAMES (Jim James)
   Started by user JAMES (Jim James)
   Created on  9-NOV-1989 09:59:29.71
   Queued on  17-NOV-1858 00:00:00.00
   Started on  9-NOV-1989 10:00:18.72
   Stopped on  9-NOV-1989 10:00:35.15
   Build job is performing a full build
   Build job has been started 1 times
   Number of changed modules scanned for build: 17
   Number of target modules for build: 1
   Build-step delay interval:     0 00:00:05.00
   Build-job timeout interval:    0 01:00:00.00
VDE>
```

This example shows the full output of the SHOW BUILD_JOB command for the build job used in the previous example.

4

```
VDE> SHOW BUILD_JOB/QUEUE V2.0-3/IDENT=2
Build job 2 for stream V2.0-3 is not yet queued for execution
   Batch queues to run 20 processes for this build job:
      Submit 3 processes (scaled from 1) on queue QUE_A
      Submit 7 processes (scaled from 3) on queue QUE_B
      Submit 2 processes (scaled from 1) on queue QUE_C
      Submit 8 processes (scaled from 4) on queue QUE_D

VDE>
```

This example displays build job 2 (specified by the /IDENTIFICATION qualifier) for stream V2.0-3. The /QUEUE qualifier displays the names

and process counts of the batch queues used to run the build job. A total of 20 processes will run the build job, distributed over four batch queues.

5
```
VDE> SHOW BUILD_JOB BL22/STEP
Build job 11 for stream BL22 is not yet queued for execution
   Build steps that are part of the build job:
      Compile step for [CODE]FOO.MAR is not started
      Link step for [CODE]FOO.EXE is not started
   Number of build steps displayed is 2

VDE>
```

This example displays information for the most recent build job in stream BL22. The /STEP qualifier displays build step information. Two steps are part of this build job: a compile step for module FOO.MAR and a link step for module FOO.EXE. The number of build steps is also shown.

6
```
VDE> SHOW BUILD_JOB/TOTAL */IDENTIFICATION=*
Number of build jobs for stream *: 14
VDE>
```

The /TOTAL qualifier displays the number of build jobs that match the selection criteria, in this example, all build jobs for all streams. There are 14 such build jobs in the VDE database.

SHOW CONTEXT

Displays the names and definitions of one or more specified contexts. A context can be used to specify the default development stream, default architecture and default facility for a VDE session.

FORMAT

SHOW CONTEXT *[ctx-name [, ctx-name...]]*

PARAMETER

ctx-name

The name of a context whose attributes are to be displayed.

You can display more than one context using wildcard characters. The percent sign (%) in a context name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Those contexts whose names match the wildcard pattern are displayed.

If no parameter is specified for the command, VDE displays the definitions of all contexts to which you have access. These include all public contexts and all of your private contexts.

DESCRIPTION

Depending on the qualifiers you select, the SHOW CONTEXT command can display the following: the name only of each selected context; the name and the default settings associated with the context; or all information about the context. The command also can show the total number of contexts that match a context name specification.

QUALIFIERS

/ALL (default)
/PRIVATE
/PUBLIC

Controls whether public, private, or all contexts are displayed. The /PRIVATE qualifier causes VDE to display only your private contexts that match the context-name parameter. The /PUBLIC qualifier causes VDE to display only the public contexts that match the parameter. The /ALL qualifier causes VDE to display both your private contexts and the public contexts that match the parameter.

Of /ALL, /PRIVATE and /PUBLIC, /ALL is the default.

/BRIEF
/NORMAL (default)
/FULL

Determines the information that is displayed about each context. The /BRIEF qualifier lists only the type (public or private) and name of each context. The /NORMAL qualifier displays the name, type, and default settings associated with each context. The /FULL qualifier lists the same information as the /NORMAL qualifier and also the creation date of

the context, a remark string, and the name of the user who created the context.

Of /BRIEF, /NORMAL and /FULL, /NORMAL is the default.

/TOTAL

Shows the total number of contexts that match each parameter. Other information about the individual contexts is not shown. A separate count is provided for each **ctx-name** parameter. Wildcard characters are usually used in the parameter when using this qualifier.

EXAMPLES

1

```
VDE> SHOW CONTEXT/BRIEF
Context BAR (public to all users)
Context DEFAULT (private to user SMITH)
Context DEFAULT (public to all users)
Context FOO (private to user SMITH)
Context NEWCTX (private to user SMITH)
Context PUBCTX (public to all users)
VDE>
```

This example shows the brief form of the SHOW CONTEXT output. Only the context names and types (public or private) are displayed.

2

```
VDE> SHOW CONTEXT PUBCTX
Context PUBCTX (public to all users)
   Default stream is V2.0-1
   Default facility is EXEC

VDE>
```

This example shows the normal form of the SHOW CONTEXT output. The command displays the name and type of context PUBCTX and it shows the default settings associated with the context.

3

```
VDE> SHOW CONTEXT/FULL/PRIVATE V3RMS
Context V3RMS (private to user SMITH)
   Default stream is V3.1-FT
   Default facility is RMS
   Context added to database on 19-APR-1989 09:08:20.92
   Context added to database by user SMITH (Jane Smith)
   Remark: Version 3.1 of RMS code

VDE>
```

The SHOW CONTEXT command displays the full output for private context V3RMS.

4

```
VDE> SHOW CONTEXT/TOTAL *
Number of contexts matching name *: 6
VDE>
```

The /TOTAL qualifier displays the total number of contexts in the database. Because the /ALL qualifier is specified by default, this count includes both all private contexts for the current user and all public contexts.

SHOW DEFAULTS

Displays the defaults for your current VDE session.

FORMAT SHOW DEFAULTS

PARAMETERS

None.

DESCRIPTION

The SHOW DEFAULTS command displays the name of the current VDE library, the current default directory, the current context, the default stream, and the default facility. You can establish the default stream and facility at one time using the SET CONTEXT /DEFAULT command, you can create multiple default contexts using CREATE CONTEXT and MODIFY CONTEXT, and you can set individual defaults with the commands SET STREAM and SET FACILITY.

QUALIFIERS

None.

EXAMPLES

1

```
VDE> SHOW DEFAULTS
Library LIBNAME in root directory DEV$:[PROJDIR]
Default directory is DEV$:[SMITH]
Context DEFAULT (private to user JONES)
   Default stream is V2.0-1
   Default facility is EXEC

VDE>
```

This example shows that the default stream is V2.0-1 and the current default facility is EXEC. These settings come from (and are identical to) context DEFAULT, a context private to the current user. The current VDE library and the current default directory are also displayed.

2

```
VDE> SHOW DEFAULTS
Library LIBNAME in root directory DEV$:[PROJDIR]
Default directory is DEV$:[SMITH]
Current session context:
   Default stream is V2.0-1
   Default facility is RMS

VDE>
```

This command shows that the default stream is V2.0-1 and the default facility in the current session is RMS. These settings do not come from a specific named context because individual settings have changed either since the last SET CONTEXT command or since the start of the VDE session.

SHOW DEVELOPER

Shows the current settings of the developer flags.

Use this command only under the explicit direction of, and with the assistance of, a VDE developer.

FORMAT SHOW DEVELOPER

DESCRIPTION

Under Construction...

EXAMPLES

1

```
VDE> SHOW DEVELOPER
Under Construction...
VDE>
```

SHOW DIRECTORY

Displays directory templates for the default stream or for the current VDE library's delta files.

FORMAT SHOW DIRECTORY

DESCRIPTION

The SHOW DIRECTORY command displays directory templates from the VDE database. Directory templates are defined with the SET DIRECTORY command and specify the names that VDE should assign to disk directories when creating directories for new streams or facilities. The SHOW DIRECTORY command displays the directory creation flags for the default stream, the template for the stream (or delta-file or staging-area) root directory, the template for each stream subdirectory, the template each facility root directory (if any), and the template for each facility subdirectory. You can use the /FACILITY qualifier to limit the display to those templates that apply to a specified facility.

For additional information on directory creation, and on directory symbols (*dir-symbol*) and directory symbol processing, see CANCEL DIRECTORY, SET DIRECTORY, and Script Functions.

QUALIFIERS

/BRIEF (default)
/FULL

Determines the information that is displayed for each directory template. The /BRIEF qualifier displays the directory specification and the directory symbol for each template. The /FULL qualifier lists the same information as the /BRIEF qualifier, but also shows the creation date of each directory template, the name of the user who created the template, and the template's remark string.

Of /BRIEF and /FULL, the former is the default.

/DELTA_FILES
/STAGING
/STREAM (default)

Determines whether directory templates for the library's delta files, its staging area, or the default stream are displayed. The /DELTA_FILES qualifier specifies that directory templates for the VDE library's delta files (CMS libraries) be displayed. The /STAGING qualifier specifies that the directory template for the library's staging-area root directory be displayed. (Staging areas hold files for queued replacements.) The /STREAM qualifier specifies that directory templates for the default stream be displayed.

Of /DELTA_FILES, /STAGING and /STREAM, /STREAM is the default.

/FACILITY=fac-name

Specifies that only those directory templates that apply to the facility given by the **fac-name** parameter be displayed. If this qualifier is omitted, VDE displays directory templates for all facilities.

You can display directory templates for multiple facilities by using wildcard characters in the **fac-name** parameter. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Directory templates for those facilities whose names match the wildcard pattern are displayed.

EXAMPLES

1

```
VDE> SHOW DIRECTORY
Directory-creation templates for stream V2.1-1:
   Directory creation flags:
      Do not defer creation of directory tree for substreams
      Do not prompt for root directory for substreams

   Stream root directory is [.V2_1-1] by default
      Subdirectory:   [.VDE$BLD] (symbol VDE$BLD)

   Directory templates for unspecified facilities:
      Subdirectory:   [.OBJ] (symbol OBJ)
      Subdirectory:   [.SRC] (symbol SRC)
      Subdirectory:   [.VDE$COM] (symbol VDE$COM)
      Subdirectory:   [.VDE$LOG] (symbol VDE$LOG)

   All other directories created using default rules

VDE>
```

This example shows the brief output of the SHOW DIRECTORY command. This command shows the directory templates for the default stream because the /STREAM qualifier is assumed by default.

2

```
VDE> SHOW DIRECTORY/DELTA_FILES
Directory-creation templates for delta files:
   Delta-file root directory is [.VDE$CMS] by default

   All other directories created using default rules

VDE>
```

This example shows the brief output of the SHOW DIRECTORY command when the /DELTA_FILES qualifier is specified. The command shows the directory templates for the VDE library's delta files (its CMS libraries).

```
3  VDE> SHOW DIRECTORY/FACILITY=RMS
   Directory-creation templates for stream V2.1-1:
      Directory creation flags:
         Do not defer creation of directory tree for substreams
         Do not prompt for root directory for substreams

      Stream root directory is [.V2_1-1] by default
         Subdirectory:   [.VDE$BLD] (symbol VDE$BLD)

      Directory templates for unspecified facilities:
         Subdirectory:   [.OBJ] (symbol OBJ)
         Subdirectory:   [.SRC] (symbol SRC)
         Subdirectory:   [.VDE$COM] (symbol VDE$COM)
         Subdirectory:   [.VDE$LOG] (symbol VDE$LOG)

      Directory templates for facility RMS:
         Root directory: [MYDIR.RMS]
         Subdirectory:   [.ANA] (symbol ANA_FILES)
         Subdirectory:   [.COM] (symbol COM)
         Subdirectory:   [.SRC] (symbol SRC)
         Subdirectory:   [.VDE$COM] (symbol VDE$COM)
         Subdirectory:   [.VDE$LOG] (symbol VDE$LOG)

      All other directories created using default rules

   VDE>
```

This example shows those directory templates that apply to facility RMS. Note that all generic information is shown, including the templates for the stream root directory and for unspecified facilities, before the templates specific to facility RMS are displayed. Templates specific to facilities other than RMS are not displayed. In this case, facility RMS has templates for an explicitly specified root directory and a different set of subdirectories than other facilities. VDE uses these templates to create the directory tree for facility RMS when you create new development streams.

```
4  VDE> SHOW DIRECTORY/FULL
   Directory-creation templates for stream V2.1-1:
      Directory creation flags:
         Do not defer creation of directory tree for substreams
         Do not prompt for root directory for substreams

      Stream root directory is [.V2_1-1] by default
         Subdirectory:   [.VDE$BLD] (symbol VDE$BLD)
            Template added to database on  9-DEC-1989 20:27:58.02
            Template added to database by user SMITH (Jane Smith)
            Remark:

      Directory templates for unspecified facilities:
         Subdirectory:   [.OBJ] (symbol OBJ)
            Template added to database on 11-DEC-1989 20:37:30.45
            Template added to database by user JONES (John Jones)
            Remark: Remark string shown here
         Subdirectory:   [.SRC] (symbol SRC)
            Template added to database on 12-DEC-1989 20:45:22.37
            Template added to database by user SMITH (Jane Smith)
            Remark: Another remark string
         Subdirectory:   [.VDE$COM] (symbol VDE$COM)
            Template added to database on  9-DEC-1989 20:27:58.02
            Template added to database by user SMITH (Jane Smith)
            Remark:
         Subdirectory:   [.VDE$LOG] (symbol VDE$LOG)
            Template added to database on  9-DEC-1989 20:27:58.02
            Template added to database by user SMITH (Jane Smith)
            Remark:

      All other directories created using default rules
```

```
VDE>
```

This example shows the full output of the SHOW DIRECTORY command. For each directory template, the full output shows the date and time the template was created, the username and full name of the creator, and the template's remark string. For templates that VDE supplies by default, the template creator is simply the user who created the VDE library.

SHOW EDITOR

Display what editor VDE uses for those VDE commands that invoke an editor.

FORMAT SHOW EDITOR

PARAMETERS

None.

DESCRIPTION

The SHOW EDITOR command displays the editor strings that various VDE commands use to invoke your preferred editor, either to create new files or to display existing files. When VDE invokes an editor, it spawns a DCL editing command in a subprocess. VDE forms the editing command by concatenating your read-write or read-only editor string, a space, and the name of the file to edit. The SHOW EDITOR command displays the current read-write and read-only editor strings for your VDE session. The command can also display your default editor strings for future VDE sessions.

QUALIFIERS

/BRIEF (default)
/FULL

Control whether VDE displays the default editor strings stored in the VDE database. The /BRIEF qualifier only displays the two editor strings for your current VDE session. The /FULL qualifier diplays the editor strings for your current VDE session and your default editor strings for future VDE sessions.

Of /FULL and /BRIEF, /BRIEF is the default.

EXAMPLES

1
```
VDE> SHOW EDITOR
Current editor strings:
   Read-write editor string: TPU
   Read-only editor string:  TPU/READ_ONLY
VDE>
```

This example illustrates the brief output of the SHOW EDITOR command.

2
```
VDE> SHOW EDITOR/FULL
Current editor strings:
   Read-write editor string: TPU
   Read-only editor string:  TPU/READ_ONLY
Default editor strings:
   Read-write editor string: EDIT/EDT
   Read-only editor string:  EDIT/EDT/READ_ONLY
VDE>
```

This example illustrates the full output of the SHOW EDITOR command. In this case, the editor strings for the current VDE session specify the

TPU editor, but the default editor strings for future sessions specify the EDT editor.

SHOW FACILITY

Displays the names and attributes of one or more specified facilities.

FORMAT

SHOW FACILITY *[facil-name [, facil-name...]]*

PARAMETER

facil-name

The name of a facility whose attributes are to be displayed.

You can display more than one facility using wildcard characters. The percent sign (%) in a facility name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Those facilities whose names match the wildcard pattern are displayed.

If you do not specify a facility name or use the /CURRENT qualifier, VDE displays all facilities in the current library.

DESCRIPTION

Depending on the qualifiers you select, the SHOW FACILITY command displays either the name of each selected facility, the name and remark string of each facility, or the name and all other attributes of each facility. The command can also show the total number of facilities that match a specified facility name.

QUALIFIERS

/BRIEF
/NORMAL (default)
/FULL

Determines the information that is displayed about each facility. The /BRIEF qualifier displays the name of each facility that matches the facility name parameters. The /NORMAL qualifier displays the name and remark string of each facility. The /FULL qualifier lists all information about each facility, including its name, its creation date, its remark string, and the name of the user who created the facility.

Of /BRIEF, /NORMAL and /FULL, /NORMAL is the default.

/CURRENT

Displays information for the default facility. Do not specify the **facil-name** parameter if you use this qualifier.

/OUTPUT=file-spec

Directs the printed output of this command to a specified file. The **file-spec** parameter specifies the name of the file. VDE creates a new file with that name, directs the command's print output to that file, and prints nothing on your terminal. If this qualifier is omitted, all output appears on the terminal.

/TOTAL

Shows the total number of facilities that match each parameter. Other information about the individual facilities is not shown. A separate count is provided for each **facil-name** parameter. Wildcard characters are usually used in the parameter when using this qualifier.

EXAMPLES

1

```
VDE> SHOW FACILITY
Facility CODE            "Code modules"
Facility DELETE          "Code for DELETE command"
Facility EXEC            "VMS Executive routines"
Facility RMS             "Record Management Services"
Facility SMG             "Screen Management Package"
VDE>
```

This example lists all facilities in the current VDE library, which is the default when no facility name parameter is present. The normal form of the output gives the name and remark string of each facility.

2

```
VDE> SHOW FACILITY/BRIEF
Facility CODE
Facility DELETE
Facility EXEC
Facility RMS
Facility SMG
VDE>
```

This example illustrates the brief form of the SHOW FACILITY output, which shows only the name of each facility.

3

```
VDE> SHOW FACILITY/FULL RMS
Facility RMS
   Facility added to database on  4-APR-1989 14:57:14.12
   Facility added to database by user SMITH (Jane Smith)
   Facility can be deleted
   Remark: Record Management Services

VDE>
```

This example shows the full output of the SHOW FACILITY command.

4

```
VDE> SHOW FACILITY/TOTAL %M*
Number of facilities matching name %M*: 2
VDE>
```

The /TOTAL qualifier displays the number of facilities that match the specified parameter. The parameter (%M*) matches all facility names that have an M in the second character position. Using the library in the previous example, the command finds two such facilities, RMS and SMG.

5

```
VDE> SHOW FACILITY/CURRENT
Facility CODE            "Code modules"
VDE>
```

The /CURRENT qualifier displays the name of the default facility.

SHOW FOLD

Displays the attributes of one or more specified fold records.

FORMAT

SHOW FOLD *[mod-name [, mod-name...]]*

PARAMETER

mod-name

Specifies a module for which fold records are to be displayed. The module name consists of an optional facility name enclosed in square brackets, a module name, and an optional type name preceded by a period (such as [FACIL]MOD1.MAR). If the facility name is omitted, the module is assumed to belong to the default facility. If the type name is omitted, VDE displays fold records for all source modules with the specified module name in the given facility.

You can display fold records for more than one module by using wildcard characters in any of the three components of the module name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. VDE displays fold records for all source modules whose names match the wildcard pattern.

If you omit the **mod-name** parameter, VDE displays fold records for all modules in the default facility.

DESCRIPTION

Depending on the qualifiers you select, the SHOW FOLD command displays either a one-line summary for each fold record or all attributes of each fold record. The command can also show the total number of fold records that match a specified module name. You can select the fold records to be displayed or counted with qualifiers that specify various selection criteria. These criteria include the fold record's creation date and time, its fold identifier, its associated development stream, and the user to whom it belongs. Only those fold records that satisfy all specified selection criteria are displayed or counted.

QUALIFIERS

/BEFORE=date-time

Displays only those fold records created before the specified date and time. The date and time can be stated in the standard OpenVMS date-time format or can be one of the following keywords: YESTERDAY, TODAY, or TOMORROW. If you use a space to separate the date from the time, remember to enclose the entire date-time string in double quotes. For further information about specifying OpenVMS date-time format, see the OpenVMS DCL Concepts.

/BRIEF (default)
/FULL

Determines the information that is displayed about each fold record. The /BRIEF qualifier displays a one-line summary for each fold record that

matches the module name parameters and the other selection criteria. The /FULL qualifier lists all database information for each fold record.

/OUTPUT=file-spec

Directs the printed output of this command to a specified file. The **file-spec** parameter specifies the name of the file. VDE creates a new file with that name, directs the command's print output to that file, and prints nothing on your terminal. If this qualifier is omitted, all output appears on the terminal.

/IDENTIFICATION=fold-ident

Displays only those fold records with the fold identifier given by the **fold-ident** parameter. The fold identifier is the identifier you specified or VDE assigned when you created the fold record, usually a small integer value. If this qualifier is omitted, VDE displays fold records with all fold identifiers.

/SINCE=date-time

Displays only those fold records created after the specified date and time. The date and time can be stated in the standard OpenVMS date-time format or can be one of the following keywords: YESTERDAY, TODAY, or TOMORROW. If you use a space to separate the date from the time, remember to enclose the entire date-time string in double quotes. For further information about specifying OpenVMS date-time format, see the OpenVMS DCL Concepts.

/STREAM=stream-name

Specifies the development stream for which fold records should be displayed. The **stream-name** parameter gives the name of the stream. If this qualifier is omitted, VDE displays fold records for all development streams.

/TOTAL

Shows the total number of fold records that match each module-name parameter and satisfy the other selection criteria. Other information about the individual fold records is not shown. A separate count is provided for each **mod-name** parameter. Wildcard characters are usually used in the parameter when using this qualifier.

/USERNAME[=username]

Specifies the OpenVMS username of a user whose fold records are to be shown. If this qualifier is specified, only that user's fold records are shown. If the **username** parameter is omitted, only your own fold records are shown. If the whole qualifier is omitted, fold records for all users are shown.

EXAMPLES

1

```
VDE> SHOW FOLD
Fold 1 of [CODE]X.FOR;2(2) into stream V1.0           26-JUN-1990 JONES
VDE>
```

This example shows the brief output of the SHOW FOLD command. This output gives the fold identifier, the module name, the development stream the change should be folded into, the date the fold record was created, and the username of the fold record's owner.

SHOW FOLD

```
2   VDE> SHOW FOLD/FULL
    Fold 1 of [CODE]X.FOR;2(2)
       Module to be folded from stream MAIN into stream V1.0
       Fold record created by user JONES (Jon Jones)
       Fold record created on 26-JUN-1990 12:40:21.90
       Remark: Fix access violation in sort routine

    VDE>
```

This example shows the full output of the SHOW FOLD command. This output gives all database information for the fold record.

SHOW GENERATION

Displays current, differential or historical information about one or more specified module generations.

FORMAT

SHOW GENERATION *[gen-spec [, gen-spec...]]*

PARAMETER

gen-spec

Specifies a module generation to be displayed. A generation specification consists of the following in order:

- an optional facility name enclosed in square brackets
- a module name
- an optional type name preceded by a period
- an optional generation number preceded by a semicolon

An example of a generation specification is [RMS]MOD1.MAR;5.

If you omit the facility name, module generations from the default facility are displayed. If you omit the type name, generations for all modules with the specified module name in the specified facility are displayed. If you omit the generation number, the most recent generation of the module in the specified (or default) stream is displayed.

If you specify a zero or negative generation number, that number is relative to the most recent generation of the module in the stream specified by the /STREAM qualifier (or the default stream if that qualifier is omitted). For example, if the most recent generation is generation 5, then generation 0 is the same as generation 5 and generation –1 is the same as generation 4.

If you specify the /GENERATION qualifier, VDE displays the generation with the given CMS generation expression and ignores the generation number on the **gen-spec** parameter.

You can display more than one generation by using wildcard characters in the facility name, module name, or type name, or by specifying a wildcard on the /GENERATION qualifier. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. An asterisk (*) in place of the generation number matches all generations of a module. Those module generations whose names match the wildcard pattern are displayed.

If you omit the **gen-spec** parameter, VDE displays the most recent generation of each module in the default facility.

SHOW GENERATION

DESCRIPTION

The SHOW GENERATION command displays part or all of the historical information stored in the VDE database for each selected generation. Unless you use the /GENERATION or /TREE qualifier, VDE only displays generation histories for a specified (or the default) development stream. The command can also show the total number of generations that match a generation specification or certain statistics about the generations that match a generation specification.

QUALIFIERS

/ALL
/SOURCE (default)
/DERIVED

Controls whether generations for source modules, derived modules, or all modules are displayed. The /SOURCE qualifier causes VDE to display only source module generations that match the **gen-spec** parameters. The /DERIVED qualifier causes VDE to display only derived module generations that match the parameters. The /ALL qualifier causes VDE to display both source module and derived module generations that match the parameters.

Of /ALL, /SOURCE and /DERIVED, /SOURCE is the default.

/BEFORE=date-time

Displays only those generations created before the specified date and time. The date and time can be stated in the standard OpenVMS date-time format or can be one of the following keywords: YESTERDAY, TODAY, or TOMORROW. If you use a space to separate the date from the time, remember to enclose the entire date-time string in double quotes. For further information about specifying OpenVMS date-time format, see the OpenVMS DCL Concepts.

/BRIEF (default)
/FULL

Determines the information that is displayed for each generation. The /BRIEF qualifier displays the module name, generation number, creating user, and creation date and time of each generation that matches the **gen-spec** parameter. The /FULL qualifier displays all attributes of each generation, including its name, generation number, creation date, creating user, CMS generation expression, and remark string.

Of /BRIEF and /FULL, /BRIEF is the default.

/GENERATION=gen-expr

Specifies that the generation with the CMS generation expression given by the **gen-expr** parameter be diplayed. When this qualifier is specified, the generation number specified on the **gen-spec** parameter is ignored.

/IF_SAME, /IF_DIFFERENT, /GENERATION and /TREE are mutually exclusive qualifiers. They also cause VDE to ignore any generation number you specify because they always reference the latest module generations in the specified streams.

/IF_SAME=stream-name
/IF_DIFFERENT=stream-name

Displays only those generations that are the same (or different) in two streams. These qualifiers select the latest generation of each specified module in the stream given by the /STREAM qualifier (or your default stream).

The /IF_SAME qualifier displays this generation only if it is also the latest generation of the module in the stream given by the **stream-name** parameter.

The /IF_DIFFERENT qualifier displays this generation only if it differs from the latest generation of the module in the stream given by the **stream-name** parameter. These qualifiers thus show for what modules two developments streams are still the same and for what modules the streams differ.

The latest generations of two streams are considered to be the same only if they are the identical generation with the same generation expression. VDE determines this from the contents of the VDE database, not by comparing the text of the two generations.

/IF_SAME, /IF_DIFFERENT, /GENERATION and /TREE are mutually exclusive qualifiers. They also cause VDE to ignore any generation number you specify because they always reference the latest module generations in the specified streams.

/OUTPUT=file-spec

Directs the printed output of this command to a specified file. The **file-spec** parameter specifies the name of the file. VDE creates a new file with that name, directs the command's print output to that file, and prints nothing on your terminal. If this qualifier is omitted, all output appears on the terminal.

/SINCE=date-time

Displays only those generations created after the specified date and time. The date and time can be stated in the standard OpenVMS date-time format or can be one of the following keywords: YESTERDAY, TODAY, or TOMORROW. If you use a space to separate the date from the time, remember to enclose the entire date-time string in double quotes. For further information about specifying OpenVMS date-time format, see the OpenVMS DCL Concepts.

/STATISTICS=(keyword [, keyword...])

Computes and displays certains statistics about the module generations that match the generation specification parameters and any selection qualifiers (such as /SINCE, /BEFORE, or /USERNAME) that you specify. The **keyword** parameters specify what statistics to display and how to sort them. The qualifier accepts the following keyword parameters:

- ALPHABETICALLY—Sort the display alphabetically by facility name, module name, or type name. If this keyword is omitted, VDE sorts the display in descending order by generation count.
- FACILITY—Compute and display the number of generations in each facility.
- MODULE—Compute and display the number of generations for each module.

- TYPES—Compute and display the number of generations that have each module type.

If you specify only one of these keywords, the surrounding parentheses may be omitted.

When this qualifier is used, other information about the individual generations is not shown. Use selection qualifiers and wildcard characters in the generation specification parameters to select the generation population for which you want statistics.

/STREAM=stream-name

Specifies that generations for the development stream given by the **stream-name** parameter be displayed. This qualifier thus affects the interpretation of generation numbers on the command's **gen-spec** parameter. If this qualifier is omitted, VDE displays generations for the default development stream.

/TOTAL

Shows the total number of generations that match the **gen-spec** parameters. Other information about the individual generations is not shown. A separate count is provided for each **gen-spec** parameter. Wildcard characters are usually used in the parameters when using this qualifier.

/TREE

Shows the whole generation tree for each specified module. This display shows the CMS generation expression for each generation, the date it was created, who created it, and the generation remark. It also shows for what streams the generation is the latest generation. This is the only display that shows all generations of a module regardless of what streams they belong to.

/IF_SAME, /IF_DIFFERENT, /GENERATION and /TREE are mutually exclusive qualifiers. They also cause VDE to ignore any generation number you specify because they always reference the latest module generations in the specified streams.

/USERNAME[=username]

Displays only those generations created by a specified user. The **username** parameter gives the OpenVMS username of the user whose generations should be displayed. If the **username** parameter is omitted, only generations that you created are shown. If the whole qualifier is omitted, generations created by all users are shown.

EXAMPLES

1

```
VDE> SHOW GENERATION/STREAM=BIRCH
Generation [RMS]MODULE_A.REQ;2(2)     by user SMITH        on 12-DEC-1989 08:44
Generation [RMS]MODULE_B.REQ;25(23A2) by user JONES        on  9-DEC-1989 20:34
Generation [RMS]MODULE_C.B32;14(14)   by user BROWN        on  9-DEC-1989 20:34
Generation [RMS]MODULE_D.B32;1(1)     by user MORELIKERED  on  9-DEC-1989 20:34
Generation [RMS]MODULE_E.B32;8(7E1)   by user JONES        on 29-DEC-1989 13:12
VDE>
```

This example shows the brief output of the SHOW GENERATION command. The command displays one line per generation. The /STREAM

qualifier says to display generations for stream BIRCH. Because no generation specification parameter is specified, the command shows the latest generation for that stream of each module in the default facility.

2
```
VDE> SHOW GENERATION MODULE_E.B32;*
Generation [RMS]MODULE_E.B32;8(8)     by user JONES        on 29-DEC-1989 13:12
Generation [RMS]MODULE_E.B32;7(7)     by user JONES        on 26-DEC-1989 14:00
Generation [RMS]MODULE_E.B32;6(6)     by user SMITH        on 19-DEC-1989 15:37
Generation [RMS]MODULE_E.B32;5(5)     by user PORTER       on 12-DEC-1989 13:10
Generation [RMS]MODULE_E.B32;4(4)     by user DINGDONG     on 12-DEC-1989 08:43
Generation [RMS]MODULE_E.B32;3(3)     by user SMITH        on 12-DEC-1989 08:37
Generation [RMS]MODULE_E.B32;2(2)     by user JONES        on 12-DEC-1989 08:35
Generation [RMS]MODULE_E.B32;1(1)     by user JONES        on  9-DEC-1989 20:34
VDE>
```

This example displays generations for module MODULE_E.B32 in facility RMS. It uses the asterisk wildcard character (*) in place of the generation number. As a result, VDE displays the latest generation of the module in the current default stream and all previous generations leading up to that generation of the module. This display thus summarizes the historical development of that module for the default stream.

3
```
VDE> SHOW GENERATION [RMS]MODULE_E.B32;8/FULL
Generation [RMS]MODULE_E.B32;8(8)
   Generation created on 29-DEC-1989 13:12:51.55
   Generation created by user JONES (John Jones)
   Module is a source module
   CMS generation expression is "8"
   Delta letters in use:  None
   Generation created in stream HICKORY
   Latest generation for stream HICKORY, BIRCH
   Module is current for stream HICKORY
   Main thread belongs to stream HICKORY
   Remark: Add the mumble-fratz feature

VDE>
```

This example illustrates the full output of the SHOW GENERATION command. VDE displays all attributes of the specified module generation. The generation shown is generation 8 relative to the default stream.

4
```
VDE> SHOW GENERATION/TREE [RMS]E.B32
Generation tree for module [RMS]E.B32:
   (4)      28-JUN-1990 16:15:13.95 JONES "Make comments meaningful"
               Latest generation for stream MAIN
   (3)      27-JUN-1990 14:14:21.71 BROWN "Convert to new QCB format"
   (1A1)    22-JUN-1990 14:56:28.43 ANDERSON "Fix timing bug"
               Latest generation for stream HICKORY
   (2)      22-JUN-1990 14:55:01.29 SMITH "Add widget feature"
   (1)      18-JUN-1990 17:46:47.60 SMITH "Initial generation"

VDE>
```

This example shows the whole generation tree for module E.B32 in facility RMS. Generation 4 is the latest generation in stream MAIN and generation 1A1 is the latest generation in stream HICKORY.

5
```
VDE> SHOW GENERATION/TOTAL/SOURCE [RMS]*;*
Number of generations matching name [RMS]*.*;*: 237
VDE>
```

In this example, the /TOTAL qualifier shows the total number of source-module generations in facility RMS. There are 237 such generations.

SHOW GROUP

Displays the names, members, and other attributes of one or more specified groups.

FORMAT

SHOW GROUP *[group-name [, group-name...]]*

PARAMETER

group-name

The name of a source group or build group in the default development stream whose attributes are to be displayed.

You can display more than one group using wildcard characters. The percent sign (%) in a group name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Those groups whose names match the wildcard pattern are displayed.

If no group name parameter is specified, VDE displays all group definitions for the default stream.

DESCRIPTION

Depending on the qualifiers you select, the SHOW GROUP command displays the names, members, and other attributes of the specified source groups or build groups. The /SOURCE or /BUILD qualifier determines whether the command displayes source groups or build groups. The command can also show the total number of groups that match a group-name specification. This command only displays group definitions for the default development stream.

QUALIFIERS

/BRIEF (default)
/FULL

Determines the type of information that is displayed for a group. The /BRIEF qualifier lists only the names of the groups that match the group-name parameters. The /FULL qualifier lists the name of each group and also its creation date, its remark string, the stream to which it belongs, and the name of the creator of the group definition.

Of /BRIEF and /FULL, /BRIEF is the default.

/BUILD
/SOURCE (default)

Specifies whether source groups or build groups are displayed. The /SOURCE qualifier causes VDE to display the source groups with the specified names. The /BUILD qualifier causes VDE to display the build groups with the specified names. If neither qualifier is specified, VDE displays source groups.

Of /BUILD and /SOURCE, /SOURCE is the default.

/MEMBERS
/NOMEMBERS (default)

Determines whether the members of the group are listed. The /MEMBERS qualifier prints the name of each module that is a member of the group. If the /FULL qualifier is also used, the /MEMBERS qualifier additionally shows who added the member to the group and the date the member was added. The /NOMEMBERS qualifier suppresses this output.

/TOTAL

Shows the total number of groups that match the group-name parameters. Other information about the individual groups is not shown. A separate count is provided for each **group-name** parameter. Wildcard characters are usually used in the **group-name** parameters when using this qualifier.

EXAMPLES

1

```
VDE> SHOW GROUP
Source group ALLMODS
Source group GROUP-2
Source group GRPNAM
Source group GRP_1
Source group GRP_2
Source group GRP_3
Source group GRP_4
VDE>
```

This example illustrates how the SHOW GROUP command without parameters or qualifiers lists the names of all source groups defined in the default development stream.

2

```
VDE> SHOW GROUP/BUILD BLDGRP, BLDGRP-2
Build group BLDGRP
Build group BLDGRP-2
VDE> SHOW GROUP/BUILD BLD*
Build group BLDGRP
Build group BLDGRP-2
VDE>
```

In this example, individual build-group names are used as parameters to list two build groups, both defined in the default stream. The second SHOW GROUP command illustrates the use of a wildcard character (*) to list multiple build groups.

3

```
VDE> SHOW GROUP/SOURCE ALLMODS/FULL
Source group ALLMODS
   Group added to database on  4-AUG-1988 15:15:31.49
   Group added to database by user DOE (John Doe)
   Group defined in stream MAIN
   Remark:

VDE>
```

Here the /FULL qualifier shows when source group ALLMODS was added to the database, the name of the user who added it, the name of the development stream in which the group was created, and the remark string associated with the group.

4
```
VDE> SHOW GROUP/BUILD/MEMBERS SYSGRP
Build group SYSGRP
   Members of the group:
      Module [BUILD_TEST]X.EXE
      Module [BUILD_TEST]Y.EXE
      Module [BUILD_TEST]Z.HLB

VDE>
```

Here the /MEMBERS qualifier lists the names of all modules that are members of build group SYSGRP. The facility name, module name, and module type of each member are displayed.

5
```
VDE> SHOW GROUP/BUILD/FULL/MEMBERS SYSGRP
Build group SYSGRP
   Group added to database on  4-AUG-1988 15:15:31.49
   Group added to database by user DOE (John Doe)
   Group defined in stream MAIN
   Remark: All images for conversion program
   Members of the group:
      Module [BUILD_TEST]X.EXE
         Added by user DOE on  4-AUG-1988 15:15:31.49
      Module [BUILD_TEST]Y.EXE
         Added by user DOE on  4-AUG-1988 15:15:31.49
      Module [BUILD_TEST]Z.HLB
         Added by user SMITH on 19-OCT-1988 16:10:40.05

VDE>
```

This example shows the output of the /FULL and /MEMBERS qualifiers used together. The output shows all database information about the specified group.

6
```
VDE> SHOW GROUP/TOTAL *
Number of groups matching name *: 7
VDE>
```

This example shows how the /TOTAL qualifier displays the number of source groups for the default stream that match a given group-name specification. In this case, a wildcard character (*) is specified that matches all source group names. The default stream thus has a total of seven source groups defined.

SHOW KEY

Displays the key definitions created with the DEFINE/KEY command.

FORMAT

SHOW KEY *[key-name]*

PARAMETER

key-name

The name of the key whose definition is to be displayed. For a list of valid key names, see the DEFINE/KEY command.

DESCRIPTION

The SHOW KEY command lists the key definitions created with DEFINE/KEY command. When used with the /DIRECTORY qualifier, the command lists the states associated with those key definitions. Use the SET KEY command to change key states.

QUALIFIERS

/ALL

Displays all of the key definitions for the current state or for the states specified with the /STATE qualifier. If you use the /ALL qualifier, do not specify a key name.

/BRIEF (default)
/NOBRIEF

The /BRIEF qualifier displays the key name, key definition, and state associated with the specified key. The /NOBRIEF qualifier displays any qualifiers associated with the key definition. The /BRIEF and /NOFULL qualifiers are equivalent.

/DIRECTORY

Displays the names of all the states for which keys have been defined. If you have not defined any keys, SHOW KEY/DIRECTORY displays the DEFAULT, GOLD, and BLUE states (the states used by the initial key definitions on the VDE keypad).

You cannot use the /DIRECTORY qualifier with any other SHOW KEY qualifiers. The key-name parameter is ignored if you use the /DIRECTORY qualifier.

/FULL
/NOFULL (default)

The /FULL qualifier displays all qualifiers associated with a key definition. By default, only the name, definition, and state of the key are displayed. The /FULL and /NOBRIEF qualifiers are equivalent.

/STATE=(state-name[,...])
/NOSTATE (default)

Displays the key definitions associated with the specified state or states. If you specify only one state name, you can omit the parentheses. State

names can be any alphanumeric string. State names are created with the DEFINE/KEY command.

If you omit the /STATE qualifier or use /NOSTATE, key definitions for the current state are displayed.

EXAMPLES

1

```
VDE> SHOW KEY KP4/FULL
DEFAULT keypad definitions:
  KP4 = "SHOW DEFAULTS" (echo,terminate,nolock)
```

In this example, the SHOW KEY command displays the complete definition of the KP4 key. Using the /FULL qualifier lists the qualifiers (/ECHO, /TERMINATE, and /NOLOCK) entered, either explicitly or by default, with the SHOW DEFAULTS command when the KP4 key was defined.

2

```
VDE> DEFINE/KEY/TERMINATE COMMA "SHOW LIBRARY"
%VDE-I-DEFKEY, DEFAULT key COMMA has been defined
VDE> SHOW KEY COMMA
DEFAULT keypad definitions:
  COMMA = "SHOW LIBRARY"
VDE> SHOW KEY/FULL COMMA
DEFAULT keypad definitions:
  COMMA = "SHOW LIBRARY"  (echo,terminate,nolock)
```

In this example, the first SHOW KEY command is the default version of the command: it displays both the definition of the COMMA key (SHOW LIBRARY) and the state in which it is defined (DEFAULT). The second SHOW KEY command (with the /FULL qualifier) additionally displays all qualifiers associated with the key definition, which in this example are /TERMINATE, /ECHO, and /NOLOCK; the qualifiers either can be explicitly stated when the key is defined (as /TERMINATE) or can be defaults (as /ECHO and /NOLOCK) of the DEFINE/KEY command.

3

```
VDE> SHOW KEY/ALL
DEFAULT keypad definitions:
  PF1 = "" (echo,noterminate,nolock,state=GOLD)
  PF2 = "Help" (echo,terminate,nolock)
  PF3 = "SET VERIFY" (echo,terminate,nolock)
  PF4 = "" (echo,noterminate,nolock,state=BLUE)
  KP4 = "SHOW DEFAULTS" (echo,terminate,nolock)
  COMMA = "SET CONTEXT BUGFIX" (echo,terminate,nolock)
```

This example the SHOW KEY command displays all of the key definitions that you created with the DEFINE/KEY command.

SHOW LIBRARY

Displays the attributes of the current VDE library.

FORMAT SHOW LIBRARY

PARAMETERS *None.*

DESCRIPTION

The SHOW LIBRARY command displays the name and root directory of the current library and optionally shows information from the database-header record of the library. This additional information includes the creation date of the library, a remark string, the name of the user who created the library, and the standard authorized and default privileges for new users of the library.

QUALIFIERS

/BRIEF (default)
/FULL

Determines what information about the library is displayed. The /BRIEF qualifier lists the name and the root directory of the library only. The /FULL qualifier lists the name, the root directory, and all information available from the database-header record of the library.

Of /BRIEF and /FULL, /BRIEF is the default.

EXAMPLES

1

```
VDE> SHOW LIBRARY
Library LIBNAME in root directory DEV$:[PROJDIR]
VDE>
```

This example illustrates the brief form of the SHOW LIBRARY output.

2

```
VDE> SHOW LIBRARY/FULL
Library LIBNAME in root directory DEV$:[PROJDIR]
   Created by user SMITH on  4-APR-1989 14:52:58.62
   Remark: Test library
   Created using VDE Version X0.1-1
   Automatic library conversion enabled
   Initial default privileges for new users:
      CREMOD CRESTEP RESREP
   Initial authorized privileges for new users:
      CREMOD CRESTEP CREUSR RESREP
VDE>
```

This example shows the full form of the SHOW LIBRARY output.

SHOW LOCATION

Displays the names of the disk directories that contain either the files of the default development stream, the delta files of the current library, or the staging areas for queued replacements.

FORMAT SHOW LOCATION

PARAMETERS

None.

DESCRIPTION

The SHOW LOCATION command displays the locations of the disk directories that contain the modules that make up your software system. You can display the directory locations of all files in the default development stream, of the current library's delta files (stored in the form of CMS libraries), or of the queued replacement ''staging area.''

The directories for the default development stream contain derived files and clear copies of source files. VDE uses these directories when it builds your system. By default, the SHOW LOCATION command displays the names of all directories for the default stream.

Each source module in your software system has a delta file. A delta file is a compact representation for storing all past and present generations of a source module. A delta file is the same as a CMS element. The SHOW LOCATION command can display the names of all delta-file directories.

Each facility has a separate disk directory tree for both the default stream and the delta files. You can specify that you want to display only those directories for a specific facility.

The VDE library also has a root directory for queued-replacement staging areas. Each time a user queues modules for replacement, VDE creates a new subdirectory (called a staging area) for that queued replacement and copies the modules to be replaced to the staging area. The SHOW LOCATION command can show the root directory for these staging areas.

QUALIFIERS

/DELTA_FILES
/STAGING
/STREAM (default)

Specifies whether to display directory locations for the VDE library's delta-files, the library's staging area, or the default development stream. The /DELTA_FILES qualifier displays directory locations of the library's delta-files (its CMS libraries). The /STAGING qualifier displays the root directory of the library's staging areas for queued replacements. The /STREAM qualifier displays directory locations for the default development stream.

Of /DELTA_FILES, /STAGING and /STREAM, /STREAM is the default.

/FACILITY=facil-name

Specifies that only those directory locations that hold modules for the indicated facility be displayed. If this qualifier is omitted, directory locations for all facilities are displayed.

EXAMPLES

1

```
VDE> SHOW LOCATION/FACILITY=RMS
Directory locations for stream V2.0:
   Stream root directory: DEV$:[PROJDIR.V2_0]
      Subdirectory:   DEV$:[PROJDIR.V2_0.VDE$BLD] (symbol VDE$BLD)

   Directory locations for facility RMS:
      Root directory: DEV$:[PROJDIR.V2_0.RMS]
      Subdirectory:   DEV$:[PROJDIR.V2_0.RMS.OBJ] (symbol OBJ)
      Subdirectory:   DEV$:[PROJDIR.V2_0.RMS.SRC] (symbol SRC)
      Subdirectory:   DEV$:[PROJDIR.V2_0.RMS.VDE$COM] (symbol VDE$COM)
      Subdirectory:   DEV$:[PROJDIR.V2_0.RMS.VDE$LOG] (symbol VDE$LOG)

VDE>
```

This example displays the disk directory locations for facility RMS in the default stream (stream V2.0).

2

```
VDE> SHOW LOCATION/FACILITY=RMS/DELTA_FILES
Directory locations for delta files:
   Delta-file root directory: DEV$:[PROJDIR.VDE$CMS]

   Directory locations for facility RMS:
      Root directory: DEV$:[PROJDIR.VDE$CMS.RMS]

VDE>
```

This example displays the disk directory locations for the delta-files for facility RMS.

SHOW LOG

Shows whether VDE input and output is recorded in a log file. If logging is enabled, the name of the log file is displayed. To enable logging, use the SET LOG command.

FORMAT SHOW LOG

PARAMETERS

None.

QUALIFIERS

None.

EXAMPLES

1

```
VDE> SHOW LOG
Output logging on file: DEV$:[MYDIR]VDELOGFILE.LOG;1
VDE>
```

This command shows that output logging is enabled and gives the name of the log file.

2

```
VDE> SHOW LOG
No output logging
VDE>
```

This command shows that output logging is disabled.

SHOW MEMORY

Displays information about VDE's memory usage.

FORMAT SHOW MEMORY

PARAMETERS

None.

DESCRIPTION

The SHOW MEMORY command displays information about the memory pools that VDE uses. It displays information about the following memory areas:

- VDE's internal memory pool. VDE uses this memory pool for its own dynamic memory needs. VDE allocates memory in this pool primarily for command processing, but also uses it for more permanent storage, such as storing the names of the default stream and facility. The information displayed includes the total size of this memory pool, the amount of memory allocated, the amount of free memory, and other information.
- The size of the LIB$GET_VM and LIB$GET_VM_PAGE memory pools. These memory pools are managed by Run-Time Library routines and are used by CMS, Rdb, and system components such as RMS. CMS, in particular, uses substantial amounts of LIB$GET_VM_PAGE memory.
- The size of the VDE utility's address space and the amount of virtual memory by which the address space can be expanded.

If you run out of virtual memory when running VDE, the SHOW MEMORY command can be helpful in determining where memory is being consumed. It is of course also helpful to VDE developers in identifying how memory is used.

QUALIFIERS

None.

EXAMPLES

1

```
VDE> SHOW MEMORY
Memory usage statistics:
   Number of bytes in VDE memory pool:       65536
   Number of areas in VDE memory pool:           1
   Number of allocated bytes in VDE pool:     1960
   Number of allocated blocks in VDE pool:      33
   Number of free bytes in VDE pool:         63564
   Number of free blocks in VDE pool:           17
   Number of bytes in VDE pool free list:    63548
   Number of blocks in VDE pool free list:      16
   Number of bytes of temporary memory:       1264
   Number of blocks of temporary memory:        12
   Number of successful LIB$GET_VM calls:          275
   Number of successful LIB$FREE_VM calls:         141
   Number of LIB$GET_VM bytes now allocated:    567120
   Number successful LIB$GET_VM_PAGE calls:         13
   Number successful LIB$FREE_VM_PAGE calls:         0
   Number of LIB$GET_VM_PAGE pages allocated:     1836
   Number of LIB$GET_VM_PAGE bytes allocated:   940032
   First free address at end of P0 space:     0063EE00
   First free address at end of P1 space:     7FDD6200
   Number of pages available for expansion:     182835

VDE>
```

This example shows the output of the SHOW MEMORY command. The first part of the output contains information about VDE's internal memory pool. This includes the total size of the pool, the amount of memory that is allocated, the amount that is free, the length of the free list (which should be the same as the amount of free memory except for a 16-byte list head), and the amount of ''temporary memory'' allocated. Temporary memory is allocated memory that is automatically released at the end of the current command.

The middle section of the output displays information about the LIB$GET_ VM and LIB$GET_VM_PAGE memory pools. This information comes from library routine LIB$STAT_VM. Finally, the last few lines give the sizes of the VDE utility's P0 and P1 address spaces and the number of pages by which the virtual address space can still be expanded before reaching the virtual page limit for the process.

SHOW MODULE

Displays the attributes of one or more specified modules.

FORMAT

SHOW MODULE *[mod-name [, mod-name...]]*

PARAMETER

mod-name

The name of a module whose attributes are to be displayed. The module name consists of an optional facility name enclosed in square brackets, a module name, and an optional type name preceded by a period. An example is [FACIL]MOD1.PAS. If no facility name is specified, the default facility is assumed. If no type name is specified, all modules with the specified module name in the given facility are displayed.

You can display more than one module by using wildcard characters in any of the three components of the module name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Those modules whose names match the wildcard pattern are displayed.

If you omit the **mod-name** parameter, VDE displays all modules in the default facility.

DESCRIPTION

Depending on the qualifiers you select, the SHOW MODULE command displays the name of each selected module, the name and remark string of each module, or the name and all other attributes of each module. The command can also show the total number of modules that match a module name specification or certain statistics about the modules that match a module name specification.

To obtain generation-related information on one or more modules, use the SHOW GENERATION command. See SHOW GENERATION for further information.

QUALIFIERS

/ALL
/SOURCE (default)
/DERIVED

Controls whether source modules, derived modules, or all modules are displayed. The /SOURCE qualifier causes VDE to display only source modules that match the module-name parameters. The /DERIVED qualifier causes VDE to display only derived modules that match the parameters. The /ALL qualifier causes VDE to display both source and derived modules that match the parameters.

Of /ALL, /SOURCE and /DERIVED, /SOURCE is the default.

/BRIEF
/NORMAL (default)
/FULL

Determines the information that is displayed for each module. The /BRIEF qualifier displays the name of each module that matches the module-name parameter. The /NORMAL qualifier displays the name and remark string of each module. The /FULL qualifier displays all attributes of each module, including its name, creation date, history and notes attributes, and remark string. It also indicates whether the module is a source module or a derived module.

Of /BRIEF, /NORMAL and /FULL, /NORMAL is the default.

/OUTPUT=file-spec

Directs the printed output of this command to a specified file. The **file-spec** parameter specifies the name of the file. VDE creates a new file with that name, directs the command's print output to that file, and prints nothing on your terminal. If this qualifier is omitted, all output appears on the terminal.

/STATISTICS=(keyword [, keyword...])

Computes and displays certains statistics about the modules that match the module-name parameters. The **keyword** parameters specify what statistics to display and how to sort them. The qualifier accepts the following keyword parameters:

- ALPHABETICALLY—Sort the display alphabetically by facility name or type name. If this keyword is omitted, VDE sorts the display in descending order by module count.
- FACILITY—Compute and display the number of modules in each facility.
- TYPES—Compute and display the number of modules that have each module type.

If you specify only one of these keywords, the surrounding parentheses may be omitted.

When this qualifier is used, other information about the individual modules is not shown. Use wildcard characters in the module-name parameters to select the module population for which you want statistics.

/TOTAL

Shows the total number of modules that match the module-name parameters. Other information about the individual modules is not shown. A separate count is provided for each **mod-name** parameter. Wildcard characters are usually used in the module-name parameters when using this qualifier.

EXAMPLES

1

```
VDE> SHOW MODULE/SOURCE *.*
Module [CODE]FOO.BAR                    "Nonsense module"
Module [CODE]MOD1.FOR                   "First Fortran module"
Module [CODE]MOD2.FOR                   "Second Fortran module"
Module [CODE]MOD3.FOR                   "Third Fortran module"
Module [CODE]MOD4.FOR                   "Sorting routines"
Module [CODE]MOD5.FOR                   "Random scrambling algorithm"
Module [CODE]MOD6.FOR                   "Destabilization routines"
VDE>
```

This SHOW MODULE command displays the names and remark strings of all source modules in the default facility. This is an example of the normal SHOW MODULE output.

2

```
VDE> SHOW MODULE/BRIEF FOO
Module [CODE]FOO.MAR
Module [CODE]FOO.EXE
Module [CODE]FOO.LIS
Module [CODE]FOO.OBJ
VDE>
```

This example illustrates the brief form of the SHOW MODULE output. It displays the names of all modules named FOO in the default facility. Both source modules and derived modules are listed.

3

```
VDE> SHOW MODULE [CODE]MOD1.FOR/FULL
Module [CODE]MOD1.FOR
   Module added to database on  4-APR-1989 14:58:39.66
   Module added to database by user SMITH (Jane Smith)
   Module is a source module
   Module cannot be deleted
   History information is not provided by default
   History string: C  #B
   Notes information is not provided by default
   Notes string: C  #G
   Note position: 81
   Remark: Test module

VDE>
```

This example illustrates the output of the /FULL qualifier to the SHOW MODULE command. The module name is fully specified on the command in this case, including the facility name, the module name, and the type name.

4

```
VDE> SHOW MODULE/TOTAL/SOURCE [RMS]*
Number of modules matching name [RMS]*.*: 7
VDE>
```

In this example, the /TOTAL qualifier shows the total number of source modules in facility RMS.

SHOW PRIVILEGES

Displays your VDE privileges.

FORMAT SHOW PRIVILEGES

PARAMETERS

None.

DESCRIPTION

The VDE privileges you have in your current VDE session determine the VDE commands (and in some cases the command qualifiers) you are allowed to use. The SHOW PRIVILEGES command displays the privileges you have in the current session. Optionally, the command also shows your default privileges and your authorized privileges.

For a list of all privilege names and their meanings, see the SET PRIVILEGES command.

QUALIFIERS

/BRIEF (default)
/FULL

Specifies whether to show all privileges or only those set for the current session. The /BRIEF qualifier shows the privileges set for your current session. The /FULL qualifier shows the privileges set for your current session as well as your default privileges (the privileges you get automatically at the start of each VDE session) and your authorized privileges (the privileges you are allowed to set using the SET PRIVILEGES command). If you do not specify a qualifier with the SHOW PRIVILEGES command, you receive the information displayed by the /BRIEF qualifier.

Of /BRIEF and /FULL, /BRIEF is the default.

EXAMPLES

1
```
VDE> SHOW PRIVILEGES
Currently set privileges:
      BUILD CREFAC CREGRP CREMOD CRESTEP RESREP
VDE>
```

This example shows the brief form of the SHOW PRIVILEGES output.

```
2   VDE> SHOW PRIVILEGES/FULL
    Currently set privileges:
          BUILD CREFAC CREGRP CREMOD CRESTEP RESREP
    Default privileges:
          BUILD CRECTX CREFAC CREGRP CREMOD CRESTEP CREUSR CRESTRM RESREP
    Authorized privileges:
          BUILD CRECTX CREFAC CREGRP CREMOD CRESTEP CREUSR CRESTRM FROZEN
          RESREP USERNAME
    VDE>
```

This example shows the use of the /FULL qualifier with the SHOW PRIVILEGES command.

SHOW REPLACEMENT

Displays the names and attributes of one or more specified queued replacements.

FORMAT

SHOW REPLACEMENT *[rep-name [, rep-name...]]*

PARAMETER

rep-name

The name of a queued replacement whose attributes are to be displayed. You can display more than one replacement using wildcard characters in the replacement name. The percent sign (%) in a replacement name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Those replacements whose names match the wildcard pattern are displayed.

Do not specify the **rep-name** parameter if you use the /IDENTIFICATION qualifier.

If you specify neither a replacement name nor the /IDENTIFICATION qualifier, VDE displays all replacements in the VDE database.

DESCRIPTION

Depending on the qualifiers you select, the SHOW REPLACEMENT command displays either a one-line summary for each specified queued replacement or all database information for each replacement. The command can also show the total number of replacements that match a specified replacement name.

QUALIFIERS

/ACCEPTED
/REJECTED
/IN_REVIEW
/NOT_REVIEWED

Specify that only those queued replacements with certain review statuses be displayed. The /ACCEPTED qualifier displays those replacements accepted by all reviewers. The /REJECTED qualifier displays those replacements rejected by at least one reviewer. The /IN_REVIEW qualifier displays those replacements that are accepted by at least one reviewer but are not yet accepted or rejected by all reviewers. The /NOT_REVIEWED qualifier displays those replacements not accepted or rejected by any reviewers.

If one or more of these qualifiers are specified, those replacements that have one of the specified status values are displayed. These qualifiers can thus be used in combination to show replacements of several different status values. If all of these qualifiers are omitted, VDE displays all replacements regardless of status.

/BEFORE=date-time

Displays only those replacements queued before the specified date and time. The date and time can be stated in the standard OpenVMS date-time format or can be one of the following keywords: YESTERDAY, TODAY, or TOMORROW. If you use a space to separate the date from the time, remember to enclose the entire date-time string in double quotes. For further information about specifying OpenVMS date-time format, see the OpenVMS DCL Concepts.

/BRIEF (default)
/FULL
/MODULE

Determines the information that is displayed about each replacement. The /BRIEF qualifier displays a one-line summary for each replacement. The /FULL qualifier displays all available database information for each replacement. The /MODULE qualifier displays the replacement name, the staging directory, the module list, and the remark string for each replacement.

Of /BRIEF, /FULL and /MODULE, /BRIEF is the default.

/HISTORY

Specifies that replacement history be displayed. Replacement history consists of all past replacements, both immediate and queued, that have already been performed; it does not include currently queued replacements. Replacement history allows you to see a historical record of all past modifications to your software system. This information complements the historical information available from the SHOW GENERATION command. When used with the /IDENTIFICATION qualifier, the /HISTORY qualifier displays a list of all past replacements in chronological order.

If you omit the /HISTORY qualifier, VDE displays currently queued replacements only.

/MODULE

Displays all modules comprising the replacement.

/IDENTIFICATION=id-range

Specifies that the replacements with the unique Replacement Ids given by the **id-range** parameter be displayed. Do not specify the **rep-name** parameter when you use this qualifier.

VDE assigns a unique Replacement Id to each immediate or queued replacement into the VDE library. VDE assigns Replacement Ids in chronological order so that the first replacement into the library has Id value 1, the next has Id value 2, and so on. When you use this qualifier, all replacements that fit your other selection criteria and that have Id values in the range given by the **id-range** parameter are displayed in ascending order by Replacement Id. This is the same as chronological order.

The **id-range** parameter can be a single integer value, a range of integers, or a wildcard character. If the parameter is a single integer (for example, /IDENT=5), VDE displays the replacement with that Replacement Id. If the parameter is a range of integers (specified as two integers separated by a colon as in /IDENT=3:5), VDE displays the replacements with Ids in that range. Finally, if the **id-range** parameter is the asterisk wildcard

character (as in /IDENT=*), VDE displays all replacements. To be displayed, each replacement must also fit all selection criteria that you specify with other qualifiers.

If you specify a zero or negative number in the **id-range** parameter, that number is relative to the largest Replacement Id value for the VDE library (the latest replacement). For example, if the latest replacement has Replacement Id 250, then /IDENT=0 is equivalent to /IDENT=250 and /IDENT=–1 is equivalent to /IDENT=249.

The /IDENTIFICATION qualifier is especially useful when used with the /HISTORY qualifier because then it displays all past replacements in chronological order.

/IN_REVIEW

Displays all replacement component modules presently undergoing review.

/NOT_REVIEWED

Displays all replacement component modules presently unreviewed.

/OUTPUT=file-spec

Directs the printed output of this command to a specified file. The **file-spec** parameter specifies the name of the file. VDE creates a new file with that name, directs the command's print output to that file, and prints nothing on your terminal. If this qualifier is omitted, all output appears on the terminal.

/REJECTED

Displays all modules comprising the replacement that have been rejected.

/REVIEWER[=username]

Specifies that only those replacements for which a given user is a reviewer be displayed. The **username** parameter gives the OpenVMS username of that reviewer. If you omit the **username** parameter, only those replacements for which you are a reviewer are displayed. If you omit the whole qualifier, replacements for all reviewers are displayed.

/SINCE=date-time

Displays only those replacements queued at or after the specified date and time. The date and time can be stated in the standard OpenVMS date-time format or can be one of the following keywords: YESTERDAY, TODAY, or TOMORROW. If you use a space to separate the date from the time, remember to enclose the entire date-time string in double quotes. For further information about specifying OpenVMS date-time format, see the OpenVMS DCL Concepts.

/STREAM[=stream-name]

Specifies that only those queued replacements that are for the development stream given by the **stream-name** parameter be displayed. If the **stream-name** parameter is omitted, replacements for the default development stream are displayed. If the whole qualifier is omitted, replacements for all streams are displayed.

/TOTAL

Shows the total number of replacements that match each parameter. Other information about the individual replacements is not shown. A

separate count is provided for each **rep-name** parameter. Wildcard characters are usually used in the parameter when using this qualifier.

/USERNAME[=username]

Specifies that only those replacements queued by a given user be displayed. The **username** parameter gives the OpenVMS username of that user. If you omit the **username** parameter, only your own replacements are displayed. If you omit the whole qualifier, replacements queued by all users are displayed.

EXAMPLES

1

```
VDE> SHOW REPLACEMENT
Replacement ANDERSON-1      BIRCH       26-JUN-1990 ANDERSON     Accepted
Replacement ANDERSON-2      BIRCH       27-JUN-1990 ANDERSON     Not reviewed
Replacement SORT_BUG        V2.0-UPD    27-JUN-1990 CHURCHILL    Accepted
Replacement SMITH-1         V2.0        27-JUN-1990 SMITH        Rejected
VDE>
```

This example shows the brief output of the SHOW REPLACEMENT command. This output shows the replacement name, the stream name, the date the replacement was queued, the username of the user who queued it, and its status.

2

```
VDE> SHOW REPLACEMENT/FULL SMITH-1
Queued replacement SMITH-1
   Replacement queued by user SMITH (Jane Smith)
   Replacement queued on 27-JUN-1990 15:10:27.20
   Replacement is for stream V2.0
   Unique replacement number is 16
   Replacement staging directory is DEV$:[LIBROOT.VDE$STAGE.VDE$STG_0.VDE$REP_16]
   Reservations in replacement:
      Reservation 1 of [BUILD_TEST]C.B32
      Reservation 1 of [BUILD_TEST]D.B32
   Replacement contains 2 reservations
   Replacement review status: Rejected
   Remark: Add mumble-fratz feature

VDE>
```

This example shows the full output of the SHOW REPLACEMENT command, which includes all available information about the replacement.

3

```
VDE> SHOW REPLACEMENT SMITH-1/MODULE
Queued replacement SMITH-1
   Replacement staging directory is DEV$:[LIBROOT.VDE$STAGE.VDE$STG_0.VDE$REP_16]
   Reservations in replacement:
      Reservation 1 of [BUILD_TEST]C.B32
      Reservation 1 of [BUILD_TEST]D.B32
   Replacement contains 2 reservations
   Remark: Add mumble-fratz feature

VDE>
```

This example illustrates the output of the /MODULE qualifier.

4

```
VDE> SHOW REPLACEMENT/TOTAL
Number of replacements matching name *: 4
VDE>
```

This example shows that the /TOTAL qualifier can display the total number of queued replacements in the database.

SHOW RESERVATION

Displays module reservations.

FORMAT

SHOW RESERVATION *[mod-name [, mod-name...]]*

PARAMETER

mod-name

The name of a source module whose reservation is to be displayed. A module name consists of an optional facility name enclosed in square brackets, a module name, and an optional type name preceded by a period (for example, [FACIL]MOD1.MAR). If the facility name is omitted, the default facility is assumed. If the type name is omitted, reservations for all modules with the specified name in the given facility are displayed.

You can display reservations for more than one module using wildcard characters in any of the three components of the module name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Reservations for the source modules whose names match the wildcard pattern are displayed.

You can also display reservations for multiple modules by specifying the name of a source group instead of a module name. Source groups are created with the CREATE GROUP command. If you specify a group name, each reservation for a module that is a member of the group is displayed.

If you omit the **mod-name** parameter, VDE displays all module reservations for the default facility.

DESCRIPTION

The SHOW RESERVATION command displays all existing reservations and queued reservation requests for a specified set of modules or for all modules in the default facility. A module reservation is created with the RESERVE command and gives you the exclusive right to modify that module. A reservation is terminated with the REPLACE or UNRESERVE command.

By default, the SHOW RESERVATION command shows reservations for all development streams, but the /STREAM qualifier lets you see only those reservations for a certain stream. Likewise, by default, the command shows reservations for all users, but the /USERNAME qualifier lets you see only those reservations that belong to a specific user, such as yourself. Other qualifiers, such as /BEFORE and /SINCE, lets you additionally restrict the set of reservations displayed.

QUALIFIERS

/BEFORE=date-time

Displays only those reservations created before the specified date and time. The date and time can be stated in the standard OpenVMS date-time format or can be one of the following keywords: YESTERDAY, TODAY, or TOMORROW. If you use a space to separate the date from the

time, remember to enclose the entire date-time string in double quotes. For further information about specifying OpenVMS date-time format, see the OpenVMS DCL Concepts.

/BRIEF (default)
/FULL

Determines the type of information displayed about each reservation. The /BRIEF qualifier displays the reservation identifier, the name of the reserved module, the reserving user, the date of the reservation, and the stream in which the module is reserved. This display normally fits on one line. The /FULL qualifier displays all database information about the reservation; this display requires several lines per reservation.

/OUTPUT=file-spec

Directs the printed output of this command to a specified file. The **file-spec** parameter specifies the name of the file. VDE creates a new file with that name, directs the command's print output to that file, and prints nothing on your terminal. If this qualifier is omitted, all output appears on the terminal.

/SESSION=session-name

Sessions are used to logically group a set of module reservations together, typically to group all modules related to a particular source code alteration or enhancement together. It allows all component modules reserved to be treated as a single entity for subsequent replacement operations. A session also allows additional modules to be reserved and incorporated into an existing session at a later time.

Displays only those reservations that are members of the reservation session given by the **session-name** parameter. If you specify this qualifier, do not specify **mod-name** parameters on the SHOW RESERVATION command.

Sessions can be manipulated via the REPLACE, RESERVE, UNRESERVE, MODIFY SESSION, MODIFY RESERVATION, CREATE MODULE, and CANCEL SESSION commands. And modules created by CREATE MODULE (on a queued-replacement stream) and reserved via RESERVE can be combined into the same session.

/SINCE=date-time

Displays only those reservations created after the specified date and time. The date and time can be stated in the standard OpenVMS date-time format or can be one of the following keywords: YESTERDAY, TODAY, or TOMORROW. If you use a space to separate the date from the time, remember to enclose the entire date-time string in double quotes. For further information about specifying OpenVMS date-time format, see the OpenVMS DCL Concepts.

/STREAM[=stream-name]

Displays only those reservations that were created for the development stream given by the **stream-name** parameter. If you omit the **stream-name** parameter, VDE only displays reservations created for your current default stream. If you omit the whole qualifier, VDE displays reservations for all streams.

/TOTAL

Shows the total number of reservations that match the specified module names and usernames. Other information about the individual reservations is not shown. A separate count is provided for each **modname** parameter.

/USERNAME[=username]

Specifies the OpenVMS username of a user whose reservations are to be shown. If this qualifier is specified, only that user's reservations are shown. If the **username** parameter is omitted, only your own reservations are shown. If the whole qualifier is omitted, reservations for all users are shown.

EXAMPLES

1

```
VDE> SHOW RESERVATION FOO.MAR
Reservation 1 of [FACIL]FOO.MAR             SMITH        23-MAR-1989 V5.4
Reservation 1 of [FACIL]FOO.MAR             JONES        24-MAR-1989 V5.5
VDE>
```

This command diplays the reservations for module FOO.MAR in the current default facility, FACIL. The output shows that users SMITH and JONES have reserved module FOO.MAR in streams V5.4 and V5.5, respectively.

2

```
VDE> SHOW RESERVATION/FULL
Reservation 1 of [FACIL]FOO.MAR
   Reservation created on 23-MAR-1989 18:09:51.80
   Reservation created by user SMITH (Jane Smith)
   Generation 29 (27B2) reserved from stream V5.4
   Expected generation expression at replacement is 27B3
   Concurrent reservations are allowed
   Remark: Fix QAR 23 again

Reservation 1 of [FACIL]MOD1.PAS
   Reservation created on 24-MAR-1989 18:50:33.18
   Reservation created by user JONES (John Jones)
   Generation 15 (15) reserved from stream V5.5
   Expected generation expression at replacement is 16
   Concurrent reservations are allowed
   Remark: Implement whistle feature

VDE>
```

This example shows the output of the /FULL qualifier. Because no module-name parameter is specified, the command displays reservations for all modules in the current default facility.

3

```
VDE> SHOW RESERVATION/USERNAME=SMITH FOO
Reservation 1 of [FACIL]FOO.MAR             SMITH        23-MAR-1989 V5.4
VDE>
```

In this example, only reservations of module FOO.MAR for user SMITH are displayed. User SMITH has one outstanding reservation (reservation 1) of this module.

SHOW SCRIPT

Displays the type, attributes, and text of one or more specified VDE scripts from the VDE database.

FORMAT SHOW SCRIPT

DESCRIPTION

Scripts are used to tailor the processing of VDE to local requirements. The description of the Script Commands (see Script Keywords, and the description of the CREATE SCRIPT command provide more complete explanations of the uses and types of scripts available, and how particular scripts are selected using the script-type qualifiers.

The SHOW SCRIPT command can be used to display the type, the script text, and various other attributes of the specified VDE script, or to display a list of of all scripts that match a particular module specification.

The /BRIEF, /FULL, and /TEXT qualifiers determine how much information is displayed by the command. The command only displays scripts that belongs to the default development stream. The /BRIEF and /FULL qaulifiers display information about the specified script, and the /TEXT qualifier is used to display the contents of the specified script.

Scripts are specified by their types and, for most types, by their module-name or facility-name templates. A compile script with the selection template [CODE]*.FOR, for example, is used to compile all modules of type FOR in facility CODE. However, if there is also a compile script with the template [CODE]X.FOR, then that more specific script is used to compile module X.FOR in facility CODE while the more generically specified script is used for all other Fortran modules in the facility.

When you specify a module-name or facility-name parameter on a script-selection qualifier, the SHOW SCRIPT command interprets wildcard characters in that parameter in one of two ways to select scripts for display. If the command uses *exact matching* to select scripts from the database, each wildcard character in the parameter must exactly match a wildcard character in a script's template for the script to be displayed. If the command uses *wildcard matching*, each wildcard character in the parameter matches any characters in a script's template.

With exact matching, the qualifier /COMPILE=[CODE]*.FOR thus only displays the one compile script with the module-name template [CODE]*.FOR. However, with wildcard matching, the same qualifier displays the two compile scripts with templates [CODE]*.FOR and [CODE]X.FOR because the wildcard character (*) in the qualifier matches both the asterisk (*) and the X in the script templates. As this example shows, exact matching can at most display one script while wildcard matching can display many scripts. Use the /EXACT_MATCH qualifier to request exact matching or the /WILDCARD_MATCH qualifier to request wildcard matching; wildcard matching is the default.

When you use exact matching, you must specify a script-type qualifier to select the script you want displayed. You must also specify the script segment number unless the default value applies.

Commands used within scripts are documented in Script Keywords, while script functions are in Script Functions, and script types are in Script Types. For other script-related commands, also see CREATE SCRIPT, DELETE SCRIPT, EXTRACT SCRIPT, INVOKE SCRIPT, and MODIFY SCRIPT.

QUALIFIERS

/ALL

Specifies that all scripts of all types for the default stream be displayed. This qualifier may only be used if wildcard matching is specified (that is, if you do not specify the /EXACT_MATCH qualifier). If no other script selection qualifier is specified, this qualifier is used by default.

/BRIEF (default)
/FULL

Determines the information that is displayed for each script. The /BRIEF qualifier displays the type of each script and the module-name or facility-name specification that the script is associated with. In addition to the information provided by the /BRIEF qualifier, the /FULL qualifier also lists the creation date of the script, the name of the user who created the script, and the script's remark string.

The /TEXT qualifier is used to display the contents of the specified script.

Of /BRIEF and /FULL, /BRIEF is the default.

/BUILD

Specifies that the default stream's BUILD script be displayed.

/COMPILE=mod-name

Specifies that the default stream's COMPILE scripts matching the specified **mod-name** specification be displayed.

/COPY=mod-name

Specifies that the default stream's COPY scripts matching the specified **mod-name** specification be displayed.

/CREATE_FACILITY

Specifies that the default stream's CREATE_FACILITY scripts be displayed.

/DELETE_GENERATION=mod-name

Specifies that the default stream's DELETE_GENERATION scripts matching the specified **mod-name** specification be displayed.

/DIFFERENCES=mod-name

Specifies that the default stream's DIFFERENCES scripts matching the specified **mod-name** specification be displayed.

/EXACT_MATCH
/WILDCARD_MATCH (default)

Specifies the matching mode to use to select scripts for display. The /EXACT_MATCH qualifier specifies that exact matching be used between the **mod-name** or **fac-name** parameter to your script-type qualifier and the selection template associated with each script. The /WILDCARD_MATCH qualifier specifies that wildcard matching be used to select scripts. With exact matching, the SHOW SCRIPT command will display at most one script. With wildcard matching, it may display many scripts.

Of /EXACT_MATCH and /WILDCARD_MATCH, the latter is the default.

/FETCH=mod-name

Specifies that the default stream's FETCH scripts matching the specified **mod-name** specification be displayed.

/INSERT_GENERATION=mod-name

Specifies that the default stream's INSERT_GENERATION scripts matching the specified **mod-name** specification be displayed.

/LINK=mod-name

Specifies that the default stream's LINK scripts matching the specified **mod-name** specification be displayed.

/NEW_STREAM=fac-name

Specifies that the default stream's NEW_STREAM scripts matching the specified **fac-name** specification be displayed.

/NOKEEP=mod-name

Specifies that the default stream's NOKEEP scripts matching the specified **mod-name** specification be displayed.

/REPLACE=mod-name

Specifies that the default stream's REPLACE scripts matching the specified **mod-name** specification be displayed.

/RESERVE=mod-name

Specifies that the default stream's RESERVE scripts matching the specified **mod-name** specification be displayed.

/SEGMENT=seg-num

Specifies the segment number of the script to display. The **seg-num** parameter specifies the integer segment number. If you omit this qualifier and request exact matching, VDE displays the script with segment number 10. If you omit this qualifier and request wildcard matching, VDE displays scripts with all segment numbers.

/SET_FACILITY=fac-name

Specifies that the default stream's SET_FACILITY scripts matching the specified **fac-name** specification be displayed.

/SET_STREAM

Specifies that the default stream's SET_STREAM script be displayed.

/STAGING=mod-name

Specifies that the default stream's STAGING scripts matching the specified **mod-name** specification be displayed.

/TEXT

Specifies that the text of the specified script be displayed.

/TOTAL

Shows the total number of scripts that match the specified script type and module-name or facility-name templates. This is the number of scripts that would be displayed if you omit the /TOTAL qualifier but leave all other qualifiers the same on the command. Other information about the individual scripts is not shown.

/UNRESERVE=mod-name

Specifies that the default stream's UNRESERVE scripts matching the specified **mod-name** specification be displayed.

EXAMPLES

1

```
VDE> SHOW SCRIPT/COMPILE=[*]*.FOR
Script to compile module [*]*.FOR (segment 10)
Script to compile module [CODE]*.FOR (segment 10)
Script to compile module [CODE]X.FOR (segment 10)
Script to compile module [ACCTNG]*.FOR (segment 9)
Script to compile module [ACCTNG]*.FOR (segment 10)
Script to compile module [ACCTNG]*.FOR (segment 20)
VDE>
```

This example shows the brief output of the SHOW SCRIPT command. Because this command uses wildcard matching by default, it displays all compile scripts whose module-name templates match the wildcarded string [*]*.FOR, including scripts with all segment numbers. In this example, there are six scripts that match these criteria.

2

```
VDE> SHOW SCRIPT/EXACT_MATCH/FULL/COMPILE=[CODE]*.FOR
Script to compile module [CODE]*.FOR (segment 10)
   Script added to database on  9-DEC-1989 20:34:25.44
   Script added to database by user JONES (John Jones)
   Remark: SCRIPT REMARK STRING

VDE>
```

This example shows the full output of the SHOW SCRIPT command. The script to compile Fortran modules in facility CODE was created by user Jones on the shown date and time. The output includes the script's remark string. Because the /EXACT_MATCH qualifier is specified, only the compile script with the exact module-name template [CODE]*.FOR is displayed.

3

```
VDE> SHOW SCRIPT/EXACT_MATCH/TEXT/NEW_STREAM=*
Script to create new stream for facility * (segment 5)
   Text of script:
     1: ! VDE script to populate the directories for a facility
     2: ! when a new stream is created.
     3: !
     4: EMIT "$ DEFINE/NOLOG OLDOBJ ", $DIR(OBJ,%CURRENT,%PARENT)
     5: EMIT "$ DEFINE/NOLOG NEWOBJ ", $DIR(OBJ,%CURRENT,%CURRENT)
     6: EMIT "$ COPY OLDOBJ:*.OLB NEWOBJ:"
     7: EMIT "$ COPY ",$DIR(SRC,%CURRENT,%PARENT),"*.H -"
     8: EMIT "       ",$DIR(SRC,%CURRENT,%CURRENT)
     9: EXIT

VDE>
```

This example displays a NEW_STREAM script for the default development stream. The script applies to all facilities (facility *) and its segment number is 5. The /TEXT qualifier displays the actual text of the script on the terminal.

4

```
VDE> SHOW SCRIPT/TOTAL
Number of scripts matching specifications: 17
VDE>
```

In this example, the /TOTAL qualifier displays the total number of scripts for the default stream. Because no script type is specified, this count includes all scripts of all types.

SHOW SESSION

Displays the names and attributes of one or more specified reservation sessions.

FORMAT

SHOW SESSION *[session-name [, session-name...]]*

PARAMETER

session-name

The name of a reservation session whose attributes are to be displayed. You can display more than one session using wildcard characters in the session name. The percent sign (%) in a session name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Those sessions whose names match the wildcard pattern are displayed.

Do not specify the **session-name** parameter if you use the /IDENTIFICATION qualifier.

If you specify neither a session name nor the /IDENTIFICATION qualifier, VDE displays all sessions in the VDE database.

If you do not specify a session name, VDE displays all sessions in the VDE database.

DESCRIPTION

Depending on the qualifiers you select, the SHOW SESSION command displays either a one-line summary for each specified reservation session or all database information for each session. The command can also show the total number of sessions that match a specified session name.

Sessions are used to group a set of module reservations together, typically to group all modules related to a particular source code alteration or enhancement together.

QUALIFIERS

/BEFORE=date-time

Displays only those sessions created before the specified date and time. The date and time can be stated in the standard OpenVMS date-time format or can be one of the following keywords: YESTERDAY, TODAY, or TOMORROW. If you use a space to separate the date from the time, remember to enclose the entire date-time string in double quotes. For further information about specifying OpenVMS date-time format, see the OpenVMS DCL Concepts.

/BRIEF (default)
/FULL

Determines the information that is displayed about each session. The /BRIEF qualifier displays a one-line summary for each session. The /FULL qualifier displays all available database information for each session.

Of /BRIEF and /FULL, the former is the default.

/IDENTIFICATION=id-range

Specifies that the sessions with the unique Session Ids given by the **id-range** parameter be displayed. Do not specify the **session-name** parameter when you use this qualifier.

VDE assigns a unique Session Id to each reservation session for the VDE library. VDE assigns Session Ids in chronological order so that the first session for the library has Id value 1, the next has Id value 2, and so on. Session Ids share the same number space as Replacement Ids, so you should expect gaps in the Session Ids. When you use this qualifier, all sessions that fit your other selection criteria and that have Id values in the range given by the **id-range** parameter are displayed in ascending order by Session Id. This is the same as chronological order.

The **id-range** parameter can be a single integer value, a range of integers, or a wildcard character. If the parameter is a single integer (for example, /IDENT=5), VDE displays the session with that Session Id. If the parameter is a range of integers (specified as two integers separated by a colon as in /IDENT=3:5), VDE displays the sessions with Ids in that range. Finally, if the **id-range** parameter is the asterisk wildcard character (as in /IDENT=*), VDE displays all sessions. To be displayed, each session must also fit all selection criteria that you specify with other qualifiers.

If you specify a zero or negative number in the **id-range** parameter, that number is relative to the largest Session Id or Replacement Id value for the VDE library (the latest session or replacement). For example, if the latest session or replacement has Id 250, then /IDENT=0 is equivalent to /IDENT=250 and /IDENT=–1 is equivalent to /IDENT=249.

/OUTPUT=file-spec

Directs the printed output of this command to a specified file. The **file-spec** parameter specifies the name of the file. VDE creates a new file with that name, directs the command's print output to that file, and prints nothing on your terminal. If this qualifier is omitted, all output appears on the terminal.

/SINCE=date-time

Displays only those sessions created at or after the specified date and time. The date and time can be stated in the standard OpenVMS date-time format or can be one of the following keywords: YESTERDAY, TODAY, or TOMORROW. If you use a space to separate the date from the time, remember to enclose the entire date-time string in double quotes. For further information about specifying OpenVMS date-time format, see the OpenVMS DCL Concepts.

/STREAM[=stream-name]

Specifies that only those reservation sessions created for the development stream given by the **stream-name** parameter be displayed. If the **stream-name** parameter is omitted, sessions for the default development stream are displayed. If the whole qualifier is omitted, sessions for all streams are displayed.

/TOTAL

Shows the total number of sessions that match each parameter. Other information about the individual sessions is not shown. A separate count is provided for each **session-name** parameter. Wildcard characters are usually used in the parameter when using this qualifier.

/USERNAME[=username]

Specifies that only those reservation sessions created by a given user be displayed. The **username** parameter gives the OpenVMS username of that user. If you omit the **username** parameter, only your own sessions are displayed. If you omit the whole qualifier, sessions for all users are displayed.

EXAMPLES

1

```
VDE> SHOW SESSION
Reservation session FIX-SORT-BUG     28-JUN-1990 ANDERSON
Reservation session SMITH-1          25-JUN-1990 SMITH
Reservation session SMITH-2          27-JUN-1990 SMITH
VDE>
```

This example shows the brief output of the SHOW SESSION command. This output shows the session name, the date the session was created, and the username of the user to whom it belongs.

2

```
VDE> SHOW SESSION/FULL FIX-SORT-BUG
Reservation session FIX-SORT-BUG
   Session created by user ANDERSON (Andy Anderson)
   Session created on 28-JUN-1990 17:49:40.71
   Session is for stream V2.0
   Unique session number is 26
   Reservations in session:
      Reservation 1 of [SYS]SORT.PAS
   Session contains 1 reservations
   Remark: Fix bug in sort routine

VDE>
```

This example shows the full output of the SHOW SESSION command, which includes all available information about the session.

3

```
VDE> SHOW SESSION/TOTAL
Number of sessions matching name *: 4
VDE>
```

This example shows that the /TOTAL qualifier can display the total number of reservation sessions in the database.

SHOW STEP

Displays build-step dependency information for the specified build steps. Build-step dependency information consists of the kind of each build step (compile, copy, or link), the name of the module that is compiled, copied, or linked by the step, and the names of the modules that are inputs to and outputs from the build step.

FORMAT

SHOW STEP *mod-name [, mod-name...]*

PARAMETER

mod-name

The name of the module that is compiled, copied, or linked by the build step to be displayed. This name consists of an optional facility name enclosed in square brackets, a module name, and an optional type name preceded by a period (for example, [FACNAM]MOD1.PAS). If no facility name is specified, the default facility is assumed. If no type name is specified, all build steps with the specified module name in the stated facility are displayed.

You can display more than one build step at a time by using wildcard characters in any of the three components of the module name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Those build steps that compile, copy, or link modules whose names match the wildcard pattern are displayed.

If no module-name parameter is specified, VDE displays all build steps that compile, copy, or link modules in the default facility.

DESCRIPTION

The SHOW STEP command displays build dependency information for the specified build steps. The command shows the kind (compile, copy, or link) of each step, the name of the module that is compiled, copied, or linked, and optionally the names of all modules that are inputs or outputs of the step. This is the build dependency information that the BUILD command uses to perform minimal system builds and to determine sequencing constraints between build steps. The SHOW STEP command always displays build-step information for the default development stream.

QUALIFIERS

/ALL (default)
/COMPILE
/COPY
/LINK

Displays compile steps, copy steps, link steps, or all steps for the specified modules. The /COMPILE qualifier displays only compile steps; the /COPY qualifier displays only copy steps; the /LINK qualifier displays only link steps; and the /ALL qualifier displays all three kinds of steps.

Of /ALL, /COMPILE, /COPY and /LINK, the former is the default.

/BRIEF (default)
/FULL

Determines the type of information that is displayed about each specified build step. The /BRIEF qualifier displays only the type of the build step (compile, copy, or link step) and the name of the module that is compiled, copied, or linked by the step. The /FULL qualifier displays the same information as the /BRIEF qualifier and also lists the names of all input and output modules for the build step. It also shows the total number of input and output modules for the step.

Of /BRIEF and /FULL, the former is the default.

/CHANGE_LEVEL

Displays ''change level'' values for the inputs and outputs of each build step if used with the /FULL qualifier. The change level of a module is an integer value that indicates how recently the module has been updated for the default stream. Source modules are updated by the REPLACE command and certain other commands that affect the latest generation of the module for the stream. (INSERT GENERATION is an example of such a command.) Derived modules are updated when rebuilt with the BUILD, COMPILE, or LINK command. Modules that have been updated more recently have higher change level values than modules that have been updated less recently.

You can use the change level values to determine whether the output modules of a build step are up-to-date with respect to the input modules. If the output modules are up-to-date, they have the same change level value as the largest change level value among the input modules. If some input module has a higher change level value than the output modules, those output modules are not up-to-date with respect to that input module. As a result, the build step needs to be performed to propagate the changes to the input module to the output modules. VDE uses this information to determine what build steps to perform for minimal system builds.

/TOTAL

Shows the total number of build steps that match each specified **mod-name** parameter. Other information about the steps is not shown. Wildcard characters are usually used in the **mod-name** parameters when using this qualifier.

EXAMPLES

1

```
VDE> SHOW STEP
Compile Step for module [CODE]FOO.MAR
Link Step for module [CODE]FOO.EXE
Link Step for module [CODE]MOD1.EXE
Compile Step for module [CODE]MOD1.FOR
Compile Step for module [CODE]MOD2.FOR
Link Step for module [CODE]MOD4.EXE
Compile Step for module [CODE]MOD4.FOR
Link Step for module [CODE]MOD6.EXE
VDE>
```

This example illustrates the SHOW STEP command without parameters or qualifiers. Because no parameter is specified, the command displays the

default form of its output: all build steps for modules in the default facility (facility CODE) are displayed.

2
```
VDE> SHOW STEP/FULL FOO/COMPILE, MOD1/LINK
Compile Step for module [CODE]FOO.MAR
   Dependency links for stream V2.1 follow
   Inputs to the build step:
      Module [CODE]FOO.MAR
   Outputs from the build step:
      Module [CODE]FOO.OBJ
   Build step has 1 inputs and 1 outputs

Link Step for module [CODE]MOD1.EXE
   Dependency links for stream V2.1 follow
   Inputs to the build step:
      Module [CODE]MOD1.OBJ
      Module [CODE]MOD2.OBJ
   Outputs from the build step:
      Module [CODE]MOD1.EXE
      Module [CODE]MOD1.MAP
   Build step has 2 inputs and 2 outputs

VDE>
```

This command displays the compile step for module FOO.MAR and the link step for module MOD1.EXE, both in the current default facility (facility CODE). Because the /FULL qualifier is specified, the command displays all dependency links for these build steps in the default stream, showing which modules are inputs to and outputs from each build step.

3
```
VDE> SHOW STEP/FULL/CHANGE_LEVEL A.REQ, D.B32
Copy Step for module [FACIL]A.REQ
   Dependency links for stream MAIN follow
   Inputs to the build step:
      Module [FACIL]A.REQ with change level 30
   Outputs from the build step:
      Module [FACIL]A.REQ with change level 30 (up-to-date)
   Build step has 1 inputs and 1 outputs

Compile Step for module [FACIL]D.B32
   Dependency links for stream MAIN follow
   Inputs to the build step:
      Module [FACIL]A.REQ with change level 30
      Module [FACIL]D.B32 with change level 20
      Module [FACIL]B.L32 with change level 39
   Outputs from the build step:
      Module [FACIL]D.$OBJ with change level 39
      Module [FACIL]D.ANA with change level 39
   Build step has 3 inputs and 2 outputs

VDE>
```

This example shows the output of the /CHANGE_LEVEL qualifier. The change level value for module A.REQ is 30, and the ''up-to-date'' tag shows that the copy step has been performed since the last time module A.REQ was modified for stream MAIN. From the perspective of the VDE builder, A.REQ is thus up-to-date for the stream and the copy step is not needed for a minimal system build. The change level values for module D.B32 shows that the outputs of the compile step (with change level value 39) are up-to-date with respect to all the inputs to the step because the inputs have change level values that are less or equal to 39. In other words, no input module has changed since the last time the step was performed for stream MAIN.

SHOW STREAM

Displays the names and attributes of one or more specified development streams.

FORMAT

SHOW STREAM *[stream-name [, stream-name...]]*

PARAMETER

stream-name

The name of a development stream whose name and other attributes are to be displayed.

You can display more than one stream using wildcard characters. The percent sign (%) in a stream name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Those streams whose names match the wildcard pattern are displayed.

If you do not specify a **stream-name** parameter, VDE displays all streams in the current library.

DESCRIPTION

Depending on the qualifiers you select, the SHOW STREAM command displays the name of each selected stream, the name and remark string of each stream, or the name and all other attributes of each stream. The command can also show the total number of streams that match a stream name specification.

QUALIFIERS

/BRIEF
/NORMAL (default)
/FULL

Determines the information that is displayed for each stream. The /BRIEF qualifier displays the name of each stream that matches the stream-name parameters. The /NORMAL qualifier displays the name and remark string of each stream. The /FULL qualifier displays the name and all other attributes of each stream.

Of /BRIEF, /NORMAL, and /FULL, /NORMAL is the default.

/DESCENDANTS

Displays all descendants of each specified stream. This qualifier displays the name of each stream created from the specified stream (that is, whose parent is the specified stream), the descendants of each such child stream, the descendants of those descendants, and so on. The display is indented so that all streams with the same parent are indented by the same amount under that parent.

/OPEN
/FROZEN
/CLOSED

Specifies that only streams with a certain status be displayed. /OPEN specifies that open streams be displayed, /FROZEN specifies that frozen streams be displayed, and /CLOSED specifies that permanently closed streams be displayed. If you specify a combination of these qualifiers, all streams with one of the specified status values are displayed. If you omit all three qualifiers, all streams are displayed regardless of status.

/OUTPUT=file-spec

Directs the printed output of this command to a specified file. The **file-spec** parameter specifies the name of the file. VDE creates a new file with that name, directs the command's print output to that file, and prints nothing on your terminal. If this qualifier is omitted, all output appears on the terminal.

/QUEUE

Dislays the default batch queues to which build jobs for the specified stream are submitted. This information includes the name of each batch queue used to run the build job, the process count for each queue, and the total process count for the build job.

/SUCCESSOR

Displays all successor chains of each specified stream. This qualifier displays lists of streams separated by arrows (–>) to show how source changes are propagated from stream to stream by the REPLACE command.

/TOTAL

Shows the total number of streams that match the stream-name parameters. Other information about the individual streams is not shown. A separate count is provided for each **stream-name** parameter. Wildcard characters are usually used in the **stream-name** parameters when using this qualifier.

EXAMPLES

1

```
VDE> SHOW STREAM
Stream BASE-LEVEL-3     "Base level 3 for version 2.1"
Stream V1.0             "Version 1.0 final release"
Stream V1.0-FT1         "Version 1 Field Test 1 release"
Stream V1.0-FT2         "Version 1 Field Test 2 release"
Stream V1.0-FT3         "Version 1 Field Test 3 release"
Stream V1.1             "Version 1.1 as finally released"
Stream V2.0             "Version 2.0 final release"
Stream V2.0-FT          "Version 2 Field Test release"
VDE>
```

This example illustrates the normal output of the SHOW STREAM command. VDE lists all streams in the current VDE library (the default when no stream name parameter is present) and displays the name and remark string for each stream.

2
```
VDE> SHOW STREAM/BRIEF
Stream BASE-LEVEL-3
Stream V1.0
Stream V1.0-FT1
Stream V1.0-FT2
Stream V1.0-FT3
Stream V1.1
Stream V2.0
Stream V2.0-FT
VDE>
```

This example shows the brief output of the SHOW STREAM command. VDE shows the name of each stream, but no more.

3
```
VDE> SHOW STREAM/FULL BASE-LEVEL-3
Stream BASE-LEVEL-3
   Stream created on  4-APR-1989 14:58:16.13
   Stream created by user SMITH (Jane Smith)
   Parent stream is V2.0
   Stream has no successors
   Stream is not part of the main development thread
   Stream cannot be deleted
   CMS class is not maintained for stream
   Replacements are immediate for stream
   Stream is open for checkins
   Default build-step delay interval:     0 00:00:03.22
   Default build-job timeout interval:    0 00:45:05.00
   Remark: Base level 3 for version 2.1

VDE>
```

This example shows the full output of the SHOW STREAM command.

4
```
VDE> SHOW STREAM/TOTAL V*
Number of streams matching name V*: 7
VDE>
```

The /TOTAL qualifier displays the number of streams that match the name specification. In this case, the name specification (V*) matches all streams whose names begin with the letter V. There are 7 such streams.

5
```
VDE> SHOW STREAM MAIN/DESCENDANTS/SUCCESSOR/QUEUE
Stream MAIN            "Main development stream"
   Descendant streams:
      MAIN (main)
         V1.0 (main)
            V1.0-1
            V1.0-2
            V1.0-3
            V2.0 (main)
               V2.1-1
               V2.1-2-BL1
            SUBDIR_TEST
         HICKORY
   Successor chains:
      MAIN -> V1.0 -> V2.0
   Default build-job batch queues to run 1 processes:
      Submit 1 processes (scaled from 1) on queue SYS$BATCH

VDE>
```

This example shows the output of the /DESCENDANT, /SUCCESSOR, and /QUEUE qualifiers. The /DESCENDANTS qualifier shows the whole tree of descendants created, directly or indirectly, from stream MAIN. The /SUCCESSOR qualifier shows that the REPLACE command will propagate changes from stream MAIN into streams V1.0 and V2.0. The

/QUEUE qualifier shows that build jobs for stream MAIN will submit one process to queue SYS$BATCH by default.

SHOW SYMBOL

Displays all symbol definitions established with the DEFINE/COMMAND command. Such symbols are used for VDE command abbreviations.

FORMAT

SHOW SYMBOL *symbol-name*

PARAMETER

symbol-name

Specifies the name of the symbol to show. If *symbol-name* includes asterisks (*), each asterisk is treated as a wildcard character that matches zero or more characters. All symbols whose names match the wildcard pattern are shown.

DESCRIPTION

The DEFINE/COMMAND command allows you to create command abbreviations in the VDE command language. The SHOW SYMBOL command lets you see all the symbols you have defined.

QUALIFIERS

None.

EXAMPLES

1

```
VDE> DEFINE/COMMAND SM = "SHOW MODULE"
VDE> SHOW SYMBOL SM
Defined commands:
  SM = "SHOW MODULE"
VDE>
```

In this example, the DEFINE/COMMAND command creates a command abbreviation that binds the string ‘‘SHOW MODULE’’ to symbol SM. The SHOW SYMBOL command then displays the command definition for SM.

2

```
VDE> SHOW SYMBOL *
Defined commands:
  SETSTRMFACIL = "SET STREAM MAIN; SET FACILITY RMS"
  SM = "SHOW MODULE"
VDE>
```

The SHOW SYMBOL command contains a wildcard character for the symbol name. As a result, the command displays the definitions of all defined symbols.

SHOW TERMINAL

Displays the line width and page size that VDE uses to format terminal output. To change these settings, use the SET TERMINAL command.

FORMAT SHOW TERMINAL

PARAMETERS

None.

QUALIFIERS

None.

EXAMPLES

1

```
VDE> SHOW TERMINAL
Terminal page size: 50, Terminal width: 80
VDE>
```

The SHOW TERMINAL command shows that VDE currently uses a page size of 50 lines and a line width of 80 columns to format output to the terminal.

SHOW USER

Displays the database records of specified users of the current VDE library.

FORMAT

SHOW USER *[username [, username...]]*

PARAMETER

username

The OpenVMS username of an existing user of the current VDE library. VDE displays this user's database record.

You can display information for many users by using wildcard characters in the username. The percent sign (%) in a username matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. VDE displays all user records whose usernames match the wildcard pattern.

If you do not specify a username parameter or the /CURRENT qualifier, VDE displays all user records in the database.

DESCRIPTION

Depending on the qualifiers you select, the SHOW USER command displays the OpenVMS username, full name string, creator and creation date of the user's database record, remark string, and authorized and default privileges for each specified user. The command also can show the total number of users that match a given username specification.

QUALIFIERS

/BRIEF (default)
/FULL

Determines the type of information about the specified users that is displayed. The /BRIEF qualifier lists the username and full name string of each user. The /FULL qualifier displays the same information as the /BRIEF qualifier and also the name of the creator of the user record and the date the record was created, the remark string, and the user's authorized and default privileges.

Of /BRIEF and /FULL, the former is the default.

/CURRENT

Displays the user record of the current user. Do not specify the **username** parameter if you use this qualifier.

/OUTPUT=file-spec

Directs the printed output of this command to a specified file. The **file-spec** parameter specifies the name of the file. VDE creates a new file with that name, directs the command's print output to that file, and prints nothing on your terminal. If this qualifier is omitted, all output appears on the terminal.

/TOTAL

Shows the total number of users that match the username parameters. Other information about the users is not shown. A separate count is provided for each **username** parameter. Wildcard characters are usually used in the **username** parameters when using this qualifier.

EXAMPLES

1

```
VDE> SHOW USER
User DOE (John Doe)
User DOLOMITE (Rock Dolomite)
User JONES (Frank Jones)
User SMITH (Jane Smith)
VDE>
```

This example illustrates the SHOW USER command without parameters or qualifiers. The command lists the usernames and full names of all users recorded in the current database.

2

```
VDE> SHOW USER SMITH, DO*
User SMITH (Jane Smith)
User DOE (John Doe)
User DOLOMITE (Rock Dolomite)
VDE>
```

In this example, the SHOW USER command with username parameters gives a brief listing for user SMITH and all users whose usernames begin with ‘‘DO’’.

3

```
VDE> SHOW USER SMITH/FULL
User SMITH (Jane Smith)
   User added to database on 20-OCT-1988 17:56:23.83
   User added to database by user DOE (John Doe)
   Remark: OpenVMS Developer
   Default privileges:
      BUILD CREFAC CREMOD
   Authorized privileges:
      BUILD CREFAC CREMOD CREUSR

VDE>
```

Here the /FULL qualifier lists all database information for user SMITH.

4

```
VDE> SHOW USER/TOTAL D*
Number of users matching name D*: 2
VDE>
```

This example shows how the /TOTAL qualifier lists the number of users whose usernames match a given name specification. In this case, a username containing a wildcard character (D*) selects all users whose usernames begin with ‘‘D.’’ The output shows that there are two such users in the database.

SHOW VERIFY

Indicates whether command input is verified (echoed) at the terminal before the input is executed. To enable verification, use the SET VERIFY command.

FORMAT SHOW VERIFY

PARAMETERS

None.

QUALIFIERS

None.

EXAMPLES

1

```
VDE> SHOW VERIFY
Not verifying commands
VDE>
```

The SHOW VERIFY command shows that command verification is disabled.

SHOW VERSION

Displays the VDE version string for the version of VDE you are running.

FORMAT SHOW VERSION

PARAMETERS

None.

QUALIFIERS

None.

VAX

EXAMPLES

1

```
VDE> SHOW VERSION
Executing VDE for OpenVMS VAX Version V1.5-0
VDE>
```

♦

The SHOW VERSION command displays what version of VDE you are running.

SPAWN

Creates a subprocess of the current process. The context of the subprocess is copied from the current process. You can use the SPAWN command to leave VDE temporarily, perform other functions (such as reading your mail), and then return to VDE.

FORMAT

SPAWN *[command]*

PARAMETER

command

Specifies the DCL command string that executes in the created subprocess. When the command completes, the subprocess terminates and control is returned to the parent process. If this parameter is not specified, a subprocess is created that transfers control to the DCL command level.

QUALIFIERS

/INPUT=filespec

Specifies an input file containing one or more DCL command strings to be executed by the spawned subprocess. If you specify a command string along with an input file, the command string is processed before the commands in the input file. After processing is complete, the subprocess is terminated.

/LOGICAL_NAMES (default)
/NOLOGICAL_NAMES

Specifies whether the logical names of the parent process are copied to the subprocess. When you do not want the subprocess to use the logical names of the parent process, enter the qualifier /NOLOGICAL_NAMES.

/OUTPUT=filespec

Identifies the output file to which the results of the SPAWN operation are written. If you omit the /OUTPUT qualifier, output is written by default to the current SYS$OUTPUT device. Whenever you use the /NOWAIT qualifier, you should also use the /OUTPUT qualifier to specify a new output destination. Doing so prevents output from being displayed on your terminal while you are entering new VDE commands.

/PROCESS=subprocess-name

Specifies the name of the subprocess. The default name of the subprocess is USERNAME_*n*.

/SYMBOLS (default)
/NOSYMBOLS

Determines whether the system passes DCL global and local symbols to the subprocess. The default is /SYMBOLS.

/WAIT (default)
/NOWAIT

Controls whether VDE waits until the subprocess is completed before allowing more VDE commands to be entered. The /NOWAIT qualifier allows you to enter new VDE commands while the specified subprocess is running. If you specify /NOWAIT, you should also use the /OUTPUT qualifier to direct the output stream to a file rather than to your terminal. Doing so prevents your terminal from being used by more than one process simultaneously.

EXAMPLES

1

```
VDE> SPAWN
$ SHOW TIME
  24-MAR-1989 16:35:14
$ LOGOUT
  Process JONES_1 logged out at 24-MAR-1989 16:35:17.37
%VDE-I-SPAWNED, subprocess spawned
%VDE-I-RETURNED, control returned to process JONES
VDE>
```

The SPAWN command without parameters creates a subprocess that prompts for DCL command input. The user enters the SHOW TIME command at the DCL prompt and the time is displayed. The user then logs out of the subprocess, which returns control to the VDE session.

2

```
VDE> SPAWN SHOW TIME
15-OCT-1988 15:34:07
%VDE-I-SPAWNED, subprocess spawned
%VDE-I-RETURNED, control returned to process JONES
VDE>
```

This example shows how to create a subprocess that executes the DCL command SHOW TIME while you are using VDE. After the command is completed, you are returned immediately to the VDE prompt.

3

```
VDE>SPAWN MAIL
MAIL> READ/NEW
   .
   .
   .
MAIL> EXIT
%VDE-I-SPAWNED, subprocess spawned
%VDE-I-RETURNED, control returned to process SMITH
VDE>
```

This example shows how to create a subprocess that invokes the OpenVMS MAIL Utility while you are in VDE. When you exit from the MAIL Utility, the subprocess disappears and the VDE> prompt returns.

4

```
VDE> SPAWN /OUTPUT=TIME.DAT SHOW TIME
%VDE-I-SPAWNED, subprocess spawned
%VDE-I-RETURNED, control returned to process SMITH
VDE> EXIT
$ TYPE TIME.DAT
15-OCT-1988 15:34:07
$
```

This example shows how to create a subprocess to invoke the DCL command SHOW TIME while you are using the VDE Utility. The

/OUTPUT qualifier specifies that the results of the SHOW TIME command be written to a file named TIME.DAT.

5
```
VDE> SPAWN /NOLOGICAL_NAMES SET HOST
_Node: MARS
    .
    .
    .
$ LOGOUT
CRAMMER logged out at ...

%REM-S-END, control returned to node _BETA::
%VDE-I-SPAWNED, subprocess spawned
%VDE-I-RETURNED, control returned to process JONES
VDE>
```

This example shows how to use the SPAWN command to create a subprocess which uses the SET HOST command to log onto another node. To leave node MARS and move back to node BETA, enter the DCL command LOGOUT. The /NOLOGICAL_NAMES qualifier prevents the logical names of the parent process from being copied to the subprocess.

START BUILD_JOB

Starts the execution of a build job by submitting build processes to OpenVMS batch queues.

Requires BUILD privilege.

FORMAT

START BUILD_JOB *[stream-name [, stream-name...]]*

PARAMETER

stream-name

The name of a development stream. The most recent build job in this stream is started.

You can start build jobs for more than one development stream by using wildcard characters in the stream name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Build jobs are started for those streams whose names match the wildcard pattern.

If you omit the **stream-name** parameter, VDE starts the most recent build job for the default stream.

DESCRIPTION

The START BUILD_JOB command starts the most recently created build job in each stream that matches the specified stream names. (To create a build job, use the BUILD command.)

To start a build job, the START BUILD_JOB command submits one or more batch jobs to the OpenVMS batch queues associated with the build job. Then each batch job runs a program called VDE$BUILD that executes the build steps belonging to the build job. The steps are executed in a loop until all the steps are done. The VDE$BUILD program does the following for each build step:

- The program looks in the database for the next available build step. (A build step is available if it is not waiting for another step to complete.) If no step is available, the program waits a few seconds before looking for another available build step.
- When the program finds a build step to run, it looks up the VDE script for that step and, from the script, generates the DCL commands needed to run the step. The program then sends the DCL commands to a subprocess that executes the build step, and waits for the step to complete.
- After the step completes, the program records the completion status of the step in the database. A step completes either successfully or with errors. If the step completes successfully, the program updates the step's build dependency information in the database and then looks for another step to perform.

By running this program in many processes simultaneously for a single build job, you can distribute the execution of your build across several processors (whether in a single machine or in multiple machines in a cluster). The /QUEUE qualifier specifies the batch queues to run the build job. The /PROCESS_COUNT qualifier specifies the total number of processes VDE submits to the batch queues performing the build job.

If you specify the /QUEUE and /PROCESS_COUNT qualifiers with the START BUILD_JOB command, they override any previous specifications. If you do not specify them with the START BUILD_JOB command, VDE uses the queues and process count specified with the BUILD command that created the build job. If neither was specified with the BUILD command, VDE uses the default queues and process count for the stream.

You cannot start a build job that is currently running. However, you can suspend such a build job with the SUSPEND BUILD_JOB command and then restart it using the START BUILD_JOB command. (This is useful, for example, if you want to change the batch queues running the build job.) Also, you cannot start a build job that has already completed execution.

QUALIFIERS

/AFTER=date-time

Indicates that the build job begins execution after the specified date and time. The date and time are stated in standard OpenVMS date-time syntax. This qualifier specifies the start time for each of the batch jobs that execute the build job.

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each build job started. The /CONFIRM qualifier causes VDE to print a message for each build job asking whether you want that build job started. If you answer YES (or Y), the build job is started. If you answer NO (or N), the build job is not started. The /NOCONFIRM qualifier causes VDE to start the specified build jobs without asking for confirmation.

/LOG (default)
/NOLOG

Controls whether log messages are printed after each build job has started. The /LOG qualifier causes such messages to be printed and the /NOLOG qualifier suppresses them. These messages indicate that the build job has started and shows how many processes have been submitted to each batch queue associated with the build job.

/PROCESS_COUNT=proc-count

Specifies the total number of processes to execute the build job. VDE creates the processes by submitting batch jobs to the queues specified by the /QUEUE qualifier or to the default queues for the default stream. The number of processes that VDE creates (that is, the number of OpenVMS batch jobs it submits) is specified with the **proc-count** parameter. This parameter thus specifies the degree of parallelism you want for the build job.

/QUEUE=(queue-name [,queue-name...])

Specifies the OpenVMS batch queues to which the build job is submitted. Each **queue-name** parameter specifies one batch queue. The **queue-name** parameter may optionally be followed by a process count in parentheses. This count specifies the number of processes started on the queue that runs the build job. For example, SYS$BATCH(3) causes three build-job processes to be submitted to queue SYS$BATCH. If the process count is omitted, one process is submitted to the queue.

If you only specify one **queue-name** parameter, the parentheses may be omitted.

If you also use the /PROCESS_COUNT qualifier, the /PROCESS_COUNT qualifier states the total number of build-job processes to create. In that case, the process-counts attached to the **queue-name** parameters on the /QUEUE qualifier are used as scaling factors to distribute the build-job processes among the queues proportionally. For example, if you specify queues HI_BATCH, HO_BATCH(3) and a total process count of eight, then two processes are submitted to queue HI_BATCH and six processes to queue HO_BATCH. If the total process count does not divide evenly into the sum of the scaling factors, the remaining processes are allocated to the queues in the order the queues are listed. If the total process count is nine, for example, the one extra process is allocated to queue HI_BATCH because HI_BATCH is listed first.

EXAMPLES

1

```
VDE> START BUILD_JOB
%VDE-I-BLDJOBSIZ, build job 15 for stream V2.0-3 consists of 25 steps
%VDE-I-BLDJOBSTARTING, starting build job 15 for stream V2.0-3
%VDE-I-BLDJOBENT, entry number 549 submitted to queue SYS$BATCH
   Job VDEBUILD_0001 (queue CLUSTER_BATCH, entry 549) started on FOO_BATCH
%VDE-I-BLDJOBENT, entry number 550 submitted to queue SYS$BATCH
   Job VDEBUILD_0001 (queue CLUSTER_BATCH, entry 550) started on PHI_BATCH
%VDE-I-BLDJOBSTARTED, build job 15 for stream V2.0-3 started with 2 processes
VDE>
```

This command starts the most recent build job for the default development stream, in this case, build job 15 for stream V2.0-3. Two batch processes are submitted, one to batch queue FOO_BATCH and one to queue PHI_ BATCH.

2

```
VDE> START BUILD_JOB V5.3/AFTER=15:00/PROCESS_COUNT=1/QUEUE=SYS$BATCH
%VDE-I-BLDJOBSIZ, build job 12 for stream V5.3 consists of 2 steps
%VDE-I-BLDJOBSTARTING, starting build job 12 for stream V5.3
%VDE-I-BLDJOBAFTER, build job to be started after 12-JUL-1989 15:00:00.00
%VDE-I-BLDJOBENT, entry number 583 submitted to queue SYS$BATCH
   Job VDEBUILD_0001 (queue FOO_BATCH, entry 583) holding until 12-JUL-1989 15:00
%VDE-I-BLDJOBSTARTED, build job 12 for stream V5.3 started with 1 processes
VDE>
```

This example starts the most recent build job (build job 12) for stream V5.3. The /AFTER qualifier specifies that the batch job starts after 15:00 hours (3:00 pm) of the current day. The /PROCESS_COUNT and /QUEUE qualifiers specify that a single batch job is submitted to queue SYS$BATCH to run the build job. The log messages show that this happened as requested.

STOP BUILD_JOB

Stops the execution of an existing build job. A stopped build job cannot be restarted.

Requires BUILD privilege.

FORMAT

STOP BUILD_JOB *[stream-name [, stream-name...]]*

PARAMETER

stream-name

The name of a development stream. The most recent build job for this stream is stopped.

You can stop build jobs for more than one stream by using wildcard characters in the stream name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Build jobs are stopped for those streams whose names match the wildcard pattern.

If you omit the **stream-name** parameter, VDE stops the most recent build job for the default stream.

DESCRIPTION

The STOP BUILD_JOB command stops the most recently created build job for each stream that matches a specified stream name. When a build job is stopped, it is marked as stopped in the database. Any processes executing the build job stop executing as soon as they read the new job status from the database. This command stops a build job permanently: it can never be restarted. Use the STOP BUILD_JOB command to stop a running build job or to mark an existing build job as stopped (even if it has never been started). After the most recent build job for a stream is stopped, you are free to create a new build job for that stream because the stopped job cannot execute and interfere with the new job.

You cannot stop a build job that has already been stopped or that has already completed execution. Because such jobs cannot execute again, there is no need to stop them.

QUALIFIERS

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each build job stopped. The /CONFIRM qualifier causes VDE to print a message for each build job asking whether you want that build job stopped. If you answer YES (or Y), the build job is stopped. If you answer NO (or N), the build job is not stopped. The /NOCONFIRM qualifier causes VDE to stop the specified build jobs without asking for confirmation.

/LOG (default)
/NOLOG

Controls whether log messages are printed after each build job is stopped. The /LOG qualifier causes such messages to be printed and the /NOLOG qualifier suppresses them. These messages indicate that the build job has been stopped and that the database transaction has successfully committed.

EXAMPLES

1
```
VDE> STOP BUILD_JOB
%VDE-I-BLDJOBSTOP, build job 15 for stream V2.0-3 stopped
VDE>
```

This command stops the most recent build job for the default development stream. In this case, the default stream is V2.0-3 and its most recent build job is build job 15.

2
```
VDE> STOP BUILD_JOB V5.3
%VDE-I-BLDJOBSTOP, build job 12 for stream V5.3 stopped
VDE>
```

This command stops the most recent build job for specified stream V5.3.

STOP SUBPROCESS

Stops the subprocess that VDE uses to execute the DCL command files it generates from VDE scripts. VDE automatically terminates this subprocess when you exit from VDE (or from the VDE kept subprocess), but this command allows you to explicitly terminate the script subprocess in order to reduce the number of open subprocesses you have.

FORMAT STOP SUBPROCESS

PARAMETERS

None.

QUALIFIERS

None.

EXAMPLES

1

```
VDE> STOP SUBPROCESS
```

This command stops the subprocess VDE uses to execute DCL command files generated by VDE scripts.

SUSPEND BUILD_JOB

Suspends the execution of an existing build job. A suspended build job can be restarted.

Requires BUILD privilege.

FORMAT

SUSPEND BUILD_JOB *[stream-name [, stream-name...]]*

PARAMETER

stream-name

The name of a development stream. The most recent build job for this stream is suspended.

You can suspend build jobs for more than one stream by using wildcard characters in the stream name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Build jobs are suspended for those streams whose names match the wildcard pattern.

If you omit the **stream-name** parameter, VDE suspends the most recent build job for the default stream.

DESCRIPTION

The SUSPEND BUILD_JOB command suspends the most recently created build job for each stream that matches a specified stream name. When a build job is suspended, it is marked as suspended in the database. Any processes executing the build job stop executing as soon as they read the new job status from the database. A suspended build job is not permanently stopped, however: it can be restarted with a START BUILD_JOB command. Use the SUSPEND BUILD_JOB command to temporarily stop a running build job; you can restart the suspended job later or restart it on different batch queues. You should also suspend a build job if one of the nodes it is running on fails; suspending a build job and then restarting it allows the build steps that were running on the failed node to reexecute.

You can only suspend a build job that is currently queued for execution or running. There is no need to suspend a build job that has not yet been started or that has completed execution.

QUALIFIERS

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each build job suspended. The /CONFIRM qualifier causes VDE to print a message for each build job asking whether you want that build job suspended. If you answer YES (or Y), the build job is suspended. If you answer NO (or N), the build job is not suspended. The /NOCONFIRM qualifier causes VDE to suspend the specified build jobs without asking for confirmation.

/LOG (default)
/NOLOG

Controls whether log messages are printed after each build job is suspended. The /LOG qualifier causes such messages to be printed and the /NOLOG qualifier suppresses them. These messages indicate that the build job is suspended and that the database transaction has successfully committed.

EXAMPLES

1

```
VDE> SUSPEND BUILD_JOB
%VDE-I-BLDJOBSUSP, build job 15 for stream V2.0-3 suspended
VDE>
```

This command suspends the most recent build job for the default development stream, in this case, build job 15 for stream V2.0-3.

2

```
VDE> SUSPEND BUILD_JOB V5.3
%VDE-I-BLDJOBSUSP, build job 12 for stream V5.3 suspended
VDE>
```

This command suspends the most recent build job for stream V5.3. The stream name is specified in this example.

UNRESERVE

Cancels one or more module reservations in the default development stream.

Requires RESREP privilege.

FORMAT

UNRESERVE *mod-name [, mod-name...]*

PARAMETER

mod-name

Specifies one or more source modules whose reservations are to be canceled. The module name consists of an optional facility name enclosed in square brackets, a module name, and an optional type name preceded by a period (such as [FACIL]MOD1.MAR). If the facility name is omitted, the module is assumed to belong to the default facility. If the type name is omitted, all source modules with the specified module name in the given facility are unreserved.

You can cancel reservations for more than one module using wildcard characters in any of the three components of the module name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. The source modules whose names match the wildcard pattern are unreserved.

You can also unreserve multiple modules by specifying the name of a source group instead of a module name. Source groups are created with the CREATE GROUP command. If you specify a group name, each module that is a member of the group is unreserved.

DESCRIPTION

The UNRESERVE command cancels an existing reservation for each specified module. Each module must be a source module currently reserved with the RESERVE command. The module reservation is canceled only for the default development stream.

If you have more than one reservation of a module, you must specify the exact reservation to be replaced. You do this by using the /IDENTIFICATION qualifier. Use the SHOW RESERVATION command to determine the identification of each reservation.

To cancel another user's reservation, you must use the /USERNAME qualifier to specify the OpenVMS username of that other user. You must have the USERNAME privilege to use the /USERNAME qualifier.

Reservations created by the CREATE MODULE command *cannot* be canceled with the UNRESERVE command. To cancel the reservation(s) and delete these module(s), use the DELETE MODULE command.

QUALIFIERS

/CONFIRM
/NOCONFIRM (default)

Controls whether VDE asks you to confirm that you want each module unreserved. The /CONFIRM qualifier causes VDE to print a message for each module that you specify asking whether you want to unreserve that module. If you answer YES (or Y), the reservation is canceled. If you answer NO (or N), the reservation is not canceled. The /NOCONFIRM qualifier causes VDE to unreserve each specified module without asking for confirmation.

/IDENTIFICATION=res-ident

Specifies the reservation that is canceled. This qualifier is required when you have multiple reservations of the same module in the default stream. The **res-ident** parameter is the reservation identifier of the reservation to be replaced. The reservation identifier is a small integer value or the identifier you specified when reserving the module. Use the SHOW RESERVATION command to determine the reservation identifier of each reservation.

/LOG (default)
/NOLOG

Controls whether log messages are printed after each module is unreserved. The /LOG qualifier causes the messages to be printed and the /NOLOG qualifier suppresses them. The messages indicate that each module has been unreserved and that the database transaction has successfully committed.

/SESSION=session-name

Sessions are used to logically group a set of module reservations together, typically to group all modules related to a particular source code alteration or enhancement together. It allows all component modules reserved to be treated as a single entity for subsequent replacement operations. A session also allows additional modules to be reserved and incorporated into an existing session at a later time.

The /SESSION qualifier causes all module reservations comprising the **session-name** reservation session are canceled, and that the session be deleted from the VDE database. If you specify this qualifier, you cannot specify **mod-name** parameters on the UNRESERVE command.

Sessions can be manipulated via the REPLACE, RESERVE, UNRESERVE, MODIFY SESSION, MODIFY RESERVATION, CREATE MODULE, and CANCEL SESSION commands. And modules created by CREATE MODULE (on a queued-replacement stream) and reserved via RESERVE can be combined into the same session.

/STREAM=stream-name

Specifies that the modules be unreserved from the development stream given by the **stream-name** parameter. If this qualifier is omitted, the modules are unreserved from the default development stream. If this qualifier is omitted and no default stream is defined, VDE prompts you for the stream name.

/USERNAME=username

Specifies that another user's reservation is to be canceled. The **username** parameter is the OpenVMS username of the other user. You must have the USERNAME privilege to use this qualifier.

EXAMPLES

1

```
VDE> UNRESERVE [FACIL]FOO.MAR
%VDE-I-UNRESERVED, reservation for module [FACIL]FOO.MAR canceled
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This example cancels an existing reservation for module FOO.MAR in facility FACIL. The log messages show that the reservation was successfully canceled.

2

```
VDE> UNRESERVE MOD1
%VDE-I-UNRESERVED, reservation for module [COPY]MOD1.PAS canceled
%VDE-I-COMMIT, database transaction has successfully committed
VDE>
```

This command cancels the reservations for all source modules called MOD1 in the current default facility, COPY. Because there is only one such module, MOD1.PAS, the reservation for that module is canceled.

3

```
VDE> UNRESERVE/IDENTIFICATION=2/CONFIRM FOO.MAR
Unreserve module [FACIL]FOO.MAR ? [No]: YES
%VDE-I-UNRESERVED, reservation for module [FACIL]FOO.MAR canceled
%VDE-I-COMMIT, database transaction has successfully committed
```

In this example, reservation 2 of module FOO.MAR in the current default facility is canceled. The /IDENTIFICATION qualifier is required if you have more than one reservation of the same module at the same time. The /CONFIRM qualifier causes VDE to ask for confirmation before unreserving the module. In this case, the user answers ''YES'', and the reservation is canceled.

VERIFY GENERATION

Verifies that a specified set of module generations exist in the corresponding CMS libraries and optionally recovers any missing generations. This command thus ensures that CMS libraries within the VDE library are consistent with the contents of the VDE database.

FORMAT

VERIFY GENERATION *mod-name [, mod-name...]*

PARAMETER

mod-name

The name of a module whose generations are to be verified. The module name consists of an optional facility name enclosed in square brackets, a module name, and an optional type name preceded by a period. An example is [FACIL]MOD1.PAS. If no facility name is specified, the default facility is assumed. If no type name is specified, all modules with the specified module name in the given facility are verified.

You can verify generations for more than one module by using wildcard characters in any of the three components of the module name. The percent sign (%) in a name matches any single character in the position it occupies and the asterisk (*) matches zero or more characters in the position it occupies. Generations for those modules whose names match the wildcard pattern are verified.

If you omit the **mod-name** parameter, VDE verifies generations for all modules in the default facility.

DESCRIPTION

The VERIFY GENERATION command verifies that the generations you specify actually exist in the CMS libraries belonging to your VDE library. This command should be used if there is reason to think that one or more CMS libraries in the VDE library may not be consistent with the VDE database. Such a situation can arise any time you have had to restore the VDE library's database from back-up files or its CMS libraries from back-up tapes. Provided you have enabled journalling, the VDE database can be fully recovered after a failure up to and including the last completed database transaction before the failure. However, the contents of the associated CMS libraries cannot be recovered past the last available back-up. If you back up your disks daily, the CMS libraries may thus be as much as a day out of date. As a result, the CMS libraries may be missing generations created by REPLACE or PERFORM REPLACEMENT commands that were entered after that last back-up, while the VDE database contains up-to-date information about those generations.

The VERIFY GENERATION command therefore assumes that the VDE database contains correct information about what generations should be found in the CMS libraries for your VDE library. If it finds any differences between the generations that the database says ought to exist and the generations that actually exist in the CMS libraries, VDE prints a message

for each such generation. Optionally, VDE can also print a message for each generation that was correctly found.

If you specify the /RECOVER qualifier, the VERIFY GENERATION command will attempt to recover any missing generations. If the missing generations were created by REPLACE commands and if the replacements were queued, VDE retrieves the text of each missing generation from the staging area for the queued replacement that created the generation. The retrieved text is then inserted into the CMS library. For immediate replacements, VDE does not maintain staging areas, but if you have a file that contains the text of a missing generation, you can specify that the missing generation be recovered from that file. Using either the VDE staging areas or files you explicitly specify, the /RECOVER qualifier thus allows you to repair any inconsistencies between the VDE database and its CMS libraries.

The VERIFY GENERATION command is usually used with wildcard characters in the module name parameters because you normally want to verify all generations in a specific facility (if the CMS library for that facility was restored from a back-up tape) or all generations for all facilities (if your whole disk was restored). To speed up the verification, you can use the /SINCE qualifier to verify and recover only those generations created since the back-up date for the CMS libraries.

QUALIFIERS

/GENERATION=gen-expr

Specifies that the generation with the CMS generation expression given by the **gen-expr** parameter be verified. When you use this qualifier, you would normally not use wildcard characters in the module name parameter. If you specify an asterisk (*) for the **gen-expr** parameter or if you omit the entire qualifier, VDE verifies all generations for each specified module.

/LOG
/NOLOG (default)

Controls whether log messages are printed for successfully verified generations. The /LOG qualifier causes such messages to be printed and /NOLOG suppresses them. The log messages for generations that are not successfully verified are always printed and are not affected by these qualifiers.

/OUTPUT=file-spec

Directs the printed output of this command to a specified file. The **file-spec** parameter specifies the name of the file. VDE creates a new file with that name, directs the command's print output to that file, and prints nothing on your terminal. If this qualifier is omitted, all output appears on the terminal.

/RECOVER[=file-spec]

Specifies that each generation that is found in the VDE database but is missing in the corresponding CMS library should be recovered into the CMS library. If you omit the **file-spec** parameter, VDE recovers each missing generation by taking its text from the staging area for the queued replacement that created the generation, provided such a staging area exists. You should omit the **file-spec** parameter when you want to recover

all missing generations for a given set of facilities and those generations are available from staging areas.

If you specify the **file-spec** parameter, VDE recovers each missing generation by taking its text from the file specified by that parameter. If the parameter specifies a directory name without a file name, VDE assumes that the file has the same name as the module being recovered. When you use the **file-spec** parameter, you would normally also use the /GENERATION qualifier to identify the specific generation to recover and you would omit wildcard characters in the module name parameter. You should use the **file-spec** parameter when you want to recover an individual module generation that cannot be found in VDE's staging areas.

This qualifier requires the PERFREP privilege.

/SINCE=date-time

Verifies (and if requested, recovers) only those generations created on or after the specified date and time. The date and time can be stated in the standard OpenVMS date-time format or can be one of the following keywords: YESTERDAY, TODAY, or TOMORROW. If you use a space to separate the date from the time, remember to enclose the entire date-time string in double quotes. For further information about specifying OpenVMS date-time format, see the OpenVMS DCL Concepts.

EXAMPLES

1

```
VDE> VERIFY GENERATION [FACIL]*/LOG
%VDE-I-GENFOUNDCMS, generation [FACIL]A.REQ;1(1) found in CMS library
%VDE-I-GENFOUNDCMS, generation [FACIL]A.REQ;2(1A1) found in CMS library
%VDE-I-GENFOUNDCMS, generation [FACIL]A.REQ;2(2) found in CMS library
%VDE-I-GENFOUNDCMS, generation [FACIL]B.REQ;1(1) found in CMS library
%VDE-I-GENNOTFOUNDCMS, generation [FACIL]B.REQ;2(2) not found in CMS library
%VDE-I-ELENOTFOUNDCMS, element [FACIL]C.B32;1(1) not found in CMS library
%VDE-I-GENFOUNDCMS, generation [FACIL]D.B32;1(1) found in CMS library
%VDE-I-GENFOUNDCMS, generation [FACIL]E.B32;1(1) found in CMS library
%VDE-I-GENFOUNDCMS, generation [FACIL]F.B32;1(1) found in CMS library

Summary statistics:
   Number of generations successfully verified:       7
   Number of CMS elements not found:                  1
   Number of CMS generations not found:               1
   Total number of generations scanned:               9

VDE>
```

This example verifies all generations in facility FACIL to make sure that each such generation is found in the CMS library for that facility. The /LOG qualifier causes VDE to print log messages for the generations that are successfully verified as well as those that are missing. In this case, generation 2 of B.REQ is not found in the CMS library and the whole CMS element for module C.B32 is missing. The summary statistics indicate that seven generations were successfully verified and two were not.

2

```
VDE> VERIFY GENERATION/SINCE=TODAY/RECOVER [*]*
%VDE-I-GENNOTFOUNDCMS, generation [FACIL]B.REQ;2(2) not found in CMS library
%VDE-I-GENRECOVERED, generation [FACIL]B.REQ;2(2) successfully recovered
from DISK$:[LIBDIR.VDE$STAGE.VDE$STG_0.VDE$REP_3.FACIL]B.REQ;
%VDE-I-ELENOTFOUNDCMS, element [FACIL]C.B32;1(1) not found in CMS library
%VDE-I-GENNOTRECOVERED, generation [FACIL]C.B32;1(1) not recovered
-VDE-I-GENNOTQUEREPL, replacement was not queued; no staging area exists

Summary statistics:
   Number of generations successfully verified:       7
   Number of CMS elements not found:                  1
   Number of CMS generations not found:               1
   Total number of generations scanned:               9
   Number of generations recovered:                   1
   Number of generations not recovered:               1

VDE>
```

This example verifies all generations created since last midnight in all facilities. Because the /RECOVER qualifier is specified, VDE recovers all missing generations from their queued-replacement staging areas and inserts them into their CMS libraries. In this example, generation 2 of module B.REQ was successfully recovered from its staging directory. However, generation 1 of C.B32 could not be recovered because no staging area exists for the replacement that created that generation. This example illustrates a typical recovery operation after the disk containing your VDE library has been restored from back-up tapes and the VDE database has been recovered from its back-up and journal files.

3

```
VDE> VERIFY GENERATION [SYS]FOO.MAR/GENERATION=10A2/RECOVER=DISK$:[MYDIR]
%VDE-I-ELENOTFOUNDCMS, element [SYS]FOO.MAR;12(10A2) not found in CMS library
%VDE-I-GENRECOVERED, generation [SYS]FOO.MAR;12(10A2) successfully recovered
from DISK$:[MYDIR]FOO.MAR;

Summary statistics:
   Number of generations successfully verified:       0
   Number of CMS elements not found:                  1
   Number of CMS generations not found:               0
   Total number of generations scanned:               1
   Number of generations recovered:                   1
   Number of generations not recovered:               0

VDE>
```

This example verifies generation 10A2 of module FOO.MAR in facility SYS. Because that generation is missing from the CMS library in this example, VDE recovers its text from file DISK$:[MYDIR]FOO.MAR and inserts it into the CMS library for facility SYS. This example thus illustrates how you can recover a specific missing generation when the generation is not found in any VDE staging area but happens to be available elsewhere.

WAIT BUILD_JOB

Waits for an existing build job to complete.

FORMAT

WAIT BUILD_JOB *[stream-name]*

PARAMETER

stream-name

The name of a development stream. VDE waits for the most recent build job in this stream to complete. If you omit the **stream-name** parameter, VDE waits for the completion of the most recent build job in the default stream.

DESCRIPTION

The WAIT BUILD_JOB command waits for the most recently created build job in the specified stream to complete. The command checks the status of the build job in the database every 15 seconds until the build job is marked as completed. A build job is marked as completed in any of the following circumstances:

- the job completes successfully
- the job completes with errors
- the job is stopped or suspended
- the job times out (is automatically stopped) because of lack of progress

Use the WAIT BUILD_JOB command in batch jobs that must wait for the completion of a build job before another task can begin, for example, before printing the status of the job.

The WAIT BUILD_JOB command stops waiting automatically if the build job does not complete within a certain interval. You can specify that interval with the /TIMEOUT qualifier. The default timeout interval is 12 hours.

If you use the WAIT BUILD_JOB command interactively, you can abort the wait by pressing CTRL/C. Expect a delay of up to 15 seconds before CTRL/C takes effect.

QUALIFIERS

/LOG (default)
/NOLOG

Controls whether log messages are printed when VDE begins and ends the waiting period. The /LOG qualifier causes such messages to be printed and the /NOLOG qualifier suppresses them. The log message at the beginning of the waiting period indicates what build job VDE is waiting for. The log message at the end of the wait indicates that the build job completed or that the wait exceeded the timeout interval.

/TIMEOUT=time-interval

Specifies the timeout interval for the wait. If the build job does not complete in this interval, VDE ends the waiting period and prints a log message to that effect. Specify the timeout interval using the following syntax to denote the number of hours, minutes, and seconds in the interval:

hh:mm:ss.ss

If this qualifier is not specified, the command ends after 12 hours.

EXAMPLES

1

```
VDE> WAIT BUILD_JOB
Waiting until build job 34 for stream MAIN completes...
Build job 34 for stream MAIN has successfully completed
VDE>
```

This command waits for the most recent build job for the default stream to complete, in this case, build job 34 for stream MAIN. The second log message, printed after the job finished, indicates that the job completed successfully.

2

```
VDE> WAIT BUILD_JOB/TIMEOUT=5:00 V5.3
Waiting until build job 33 for stream V5.3 completes...
Timeout interval has expired (interval:    0 00:05:00.00)
VDE>
```

This command waits for the most recent build job for stream V5.3. The timeout interval is 5 minutes. In this case, the timeout interval expired before the build job completed, as indicated by the second log message.

A Associated Logical Names

This appendix describes various logical names used within the VDE package. A partial list of these logical names is present in Table A–1.

Note: This interface is under development and is subject to change. Please contact VDE support for further information.)

Table A–1 Logical Names

Logical Name	Description
VDE$LIBRARY	References the device and directory of a VDE library.
VDE$LIBRARY_DEFAULTS	A search list logical name that contains zero or more VDE libraries available by default. Used in conjunction with VDE$LIBRARY_DEFAULTS_LABELS and VDE$LIBRARY_DEFAULTS_MNEMONICS, and the number of translations must match the number of translations present in VDE$LIBRARY_DEFAULTS_LABELS and VDE$LIBRARY_DEFAULTS_MNEMONICS. If not defined or defined without translations, an attempt is made to access a list of default library labels located in the X Windows resource file.
VDE$LIBRARY_DEFAULTS_LABELS	A search list logical name that contains zero or more labels for VDE libraries. Used in conjunction with VDE$LIBRARY_DEFAULTS, and the number of translations must match the number of translations present in VDE$LIBRARY_DEFAULTS. If not defined or defined without translations, an attempt is made to access a list of default library labels in the X Windows resource file. To avoid user confusion, one should configure the remark in each library to match the associated translation(s) of the VDE$LIBRARY_DEFAULTS_LABELS logical name. See the SET LIBRARY/REMARK command.
VDE$LIBRARY_DEFAULTS_CMS	A search list logical name that contains zero or more root locations for the VDE CMS libraries. This logical is used in conjunction with the other VDE$LIBRARY_DEFAULTS logical names, and the number of translations must match the number of translations present in VDE$LIBRARY_DEFAULTS. If not defined or defined without translations, an attempt is made to access a list of default library labels in the X Windows resource file. This logical is used by a VDE manager to provide the location of the CMS libraries to programs and procedures associated with VDE, but this logical name is not used within VDE.
VDE$LIBRARY_DEFAULTS_MNEMONICS	

Table A–1 (Cont.) Logical Names

Logical Name	Description
	A search list logical name that contains zero or more mnemomic names for VDE libraries. Can be used in the SET LIBRARY command. This logical name is used in conjunction with VDE$LIBRARY_DEFAULTS, and the number of translations must match the number of translations present in VDE$LIBRARY_DEFAULTS. Mnemonics are not utilitized under the X Windows interfaces. To avoid user confusion, one should configure the name in each library to match the associated translation(s) of the VDE$LIBRARY_DEFAULTS_ MNEMONICS logical name. See the SET LIBRARY/NAME command.
VDE$LIBRARY_LABEL	Contains the remark associated with the last VDE library accessed. This logical is intended for use within command procedures calling VDE. To avoid user confusion, one should configure the remark in each library to match the associated translation(s) of the VDE$LIBRARY_DEFAULTS_ LABELS logical name. See the SET LIBRARY/REMARK command.

This logical name is write-only; VDE does not translate it and users should not modify the contents.)

VDE$LIBRARY_MNEMONIC	Contains the mnemonic, when known, or the specification associated with the last VDE library accessed. To avoid user confusion, one should configure the name in each library to match the associated translation(s) of the VDE$LIBRARY_DEFAULTS_ MNEMONICS logical name. See the SET LIBRARY/NAME command.

This logical is intended for use within command procedures calling VDE. This logical name is write-only; VDE does not translate it and users should not modify the contents.)

VDE$NOTES_LIBRARY	When defined, the translation of this logical name is used to override the default SCT notes conference specification of NOTES$LIBRARY:.NOTE. Among the obvious use in relocating the SCT notes conferences to a remote node or remote cluster, this logical name can be defined as 0::NOTES$LIBRARY:.NOTE to force all SCT note entries to made via the notes server process(es). Only one notes conference default specification is possible per node.
VDE$OFFLINE	When defined, disables VDE. The translation is assumed to contain the reason why the VDE environment is offline, and is displayed to all VDE users. Only fully-privileged definitions of this logical name are accepted and processed.
VDE$REPLACE_DEFAULT	If defined, this logical contains ‘‘EDIT’’, ‘‘FILE’’, or ‘‘NOTE’’, and selects the default processing for the location of the SCT note. If undefined or defined to an invalid value, VDE defaults to ‘‘EDIT’’, as indicated by the brackets in the ‘‘Enter file to load into SCT: (filename/[EDIT]/NOTE)’’ prompt. The brackets will indicate the default for the prompt.

not translate it and users should not modify the contents. For further information on prompting for SCT notes, see the documentation on the REPLACE command.)

Table A–1 (Cont.) Logical Names

Logical Name	Description
VDE$SCRATCH	A logical name that translates into the directory specification for the VDE scratch directory. If this logical name is not specified, a [.VDE$SCRATCH] subdirectory will be created underneath the SYS$SCRATCH: directory, and the logical name VDE$SCRATCH will be defined to reference it.
VDE$TEMPLATES	References the device and directory that contains the predefined SCT template files SCT-NEW.TEMPLATE, and SCT-BUG.TEMPLATE. This logical name is used during the REPLACE command. This logical name definition is optional—no default SCT template is available if this logical name is not defined, or if the two template files are not located. In either case, the user is placed in the chosen text editor and must include the appropriate template file manually.
VDE$WHEREIS_BACKUPS_AFTER_IMAGE	Output from the VDE/WHEREIS command. Allows a command procedure to locate the after-image BACKUP files.
VDE$WHEREIS_BACKUPS_ARCHIVE	Output from the VDE/WHEREIS command. Allows a command procedure to locate the database backup archival directory.
VDE$WHEREIS_BACKUPS_CURRENT	Output from the VDE/WHEREIS command. Allows a command procedure to locate the database backup directory containing the most recent backups.
VDE$WHEREIS_BACKUP_FULL	Output from the VDE/WHEREIS command. Allows a command procedure to retrieve the filename used for the full backups.
VDE$WHEREIS_BACKUP_INCR	Output from the VDE/WHEREIS command. Allows a command procedure to retrieve the filename used for incremental backups.
VDE$WHEREIS_BUILD	Output from the VDE/WHEREIS command. *This logical is not fully implemented.*
VDE$WHEREIS_CMS	Output from the VDE/WHEREIS command. Allows a command procedure to retrieve the root specification of the CMS library.
VDE$WHEREIS_DATABASE	Output from the VDE/WHEREIS command. Allows a command procedure to retrieve the specification of the directory containing the root database file.
VDE$WHEREIS_LABEL	Output from the VDE/WHEREIS command. Allows a command procedure to retrieve the text label associated with a particular database.
VDE$WHEREIS_MNEMONIC	

Table A–1 (Cont.) Logical Names

Logical Name	Description
	Output from the VDE/WHEREIS command. Allows a command procedure to retrieve the mnemonic name associated with a particular database.
VDE$WHEREIS_STAGING	Output from the VDE/WHEREIS command. Allows a command procedure to retrieve the root specification of the staging area for a VDE database.
VDE$WHEREIS_STATISTICS	Output from the VDE/WHEREIS command. Allows a command procedure to retrieve the specification of the statistics database file for a VDE database.

B Recording Build Dependencies

This appendix describes the callable interface that records build dependencies in the OpenVMS database. This interface allows the OpenVMS LINKER and various language processors to record the input and output modules of a given build step in the VDE database.

LIB$REC_DEPENDENCY—Recording Build Dependencies

LIB$REC_DEPENDENCY records the input and output modules for the current build step and incorporates this information into the build dependency graph in the OpenVMS database. You must make a sequence of calls on LIB$REC_DEPENDENCY to record all dependency information for the current build step.

This routine performs no action and consumes no time unless a build step is currently executing as part of a OpenVMS system build.

FORMAT

LIB$REC_DEPENDENCY *func-code, name-string, lib-index*

ARGUMENTS

func-code

type: **longword integer (signed)**
access: **read only**
mechanism: **by reference**

A function code indicating the information that is passed to LIB$REC_ DEPENDENCY. Table B–1 contains the list of defined values.

Table B–1 Dependency Codes

Value	Symbol	Description
1	VDE$K_DEPEND_BEGIN_RECORD	Begin dependency recording
2	VDE$K_DEPEND_END_RECORD	End dependency recording
3	VDE$K_DEPEND_VERSION	Record compiler or tool version
4	VDE$K_DEPEND_INPUT_FILE	Record an input file
5	VDE$K_DEPEND_INPUT_LIBRARY	Record an input library
6	VDE$K_DEPEND_INPUT_MODULE	Record an input library module
7	VDE$K_DEPEND_OUTPUT_FILE	Record an output file
8	VDE$K_DEPEND_OUTPUT_LIBRARY	Record an output library
9	VDE$K_DEPEND_OUTPUT_MODULE	Record an output library module

The meanings of the remaining parameters depend on the value of the function code.

name-string

type: **character string**
access: **read only**
mechanism: **by descriptor**

An ASCII text string whose contents depend on the function-code parameter. For the VDE$K_DEPEND_VERSION function, this string contains the name and version number of the LINKER or language

processor (for example, ''VAX Linker V5.2-1''). For the VDE$K_DEPEND_BEGIN_RECORD and VDE$K_DEPEND_END_RECORD functions, this parameter is not used. For all other function codes, this string contains the name of a file, library, or library module that is an input to or output of the current build step.

lib-index

type: **longword integer (signed)**
access: **read write**
mechanism: **by reference**

A library index whose use depends on the function-code parameter. For the VDE$K_DEPEND_INPUT_LIBRARY and VDE$K_DEPEND_OUTPUT_LIBRARY functions, this is an output parameter which receives a library index that uniquely identifies the specified input or output library (for example, an object library or a text library). For the VDE$K_DEPEND_INPUT_MODULE and VDE$K_DEPEND_OUTPUT_MODULE functions, this is an input parameter that specifies the index of the library that contains the library module specified by the **name-string** parameter. For all other function codes, this parameter is not used.

DESCRIPTION

LIB$REC_DEPENDENCY records build dependency information in the OpenVMS database for the language processor or other tool that is performing the current build step in a system build. The processor or tool (for example, the LINKER) must perform a sequence of calls on LIB$REC_DEPENDENCY with different function codes to specify all input and output modules to the current build step. The required sequence of calls is as follows:

- Make one call with the VDE$K_DEPEND_BEGIN_RECORD function code. This call begins the recording of dependency information for the build step.
- Make as many calls as necessary to specify each input file, each input library, each input library module, each output file, each output library, and each output library module for the build step. These calls may appear in any order. Duplicate calls for the same input or output entity are also allowed; LIB$REC_DEPENDENCY eliminates any duplicates before recording the information in the database.

  Also make one call with the VDE$K_DEPEND_VERSION function code. This call specifies the name and version of the compiler, LINKER, or other tool that performs the build step. This call may appear at any time with respect to the calls that specify inputs and outputs. (If several tools are used in the build step, you may make this call for each tool.)
- Make one final call with the VDE$K_DEPEND_END_RECORD function code. This call ends the dependency recording session for the build step.

Begin Recording Dependencies

To begin recording the inputs and outputs of the current build step, make a call as in the following example:

```
LIB$REC_DEPENDENCY( %REF(VDE$K_DEPEND_BEGIN_RECORD) );
```

This example (shown in Bliss) illustrates how a compiler or other tool begins recording dependency information. The function code is VDE$K_DEPEND_BEGIN_RECORD. The **name-string** and **lib-index** parameters are not used in this call.

Recording Tool Name and Version

To record the name and version number of the compiler or other tool performing the current build step, make a call as in the following example:

```
LIB$REC_DEPENDENCY( %REF(VDE$K_DEPEND_VERSION),
                    %ASCID 'VAX-11 BLISS-32 V4.2-761' );
```

This example illustrates how the Bliss compiler records its own name and version number. The function code is VDE$K_DEPEND_VERSION and the **name-string** parameter is the name and version number of the processor. The **lib-index** parameter is not used in this call.

Recording Input Files

To record a specific file as an input to the current build step, call LIB$REC_DEPENDENCY with the VDE$K_DEPEND_INPUT_FILE function code, as in the following example:

```
LIB$REC_DEPENDENCY( %REF(VDE$K_DEPEND_INPUT_FILE),
                    %ASCID 'DEVD$:[DIR.SRCDIR]MOD1.B32;3' );
```

The **name-string** parameter is the full file specification of the input file; this is the resultant name string that RMS returns. The **lib-index** parameter is not used.

Recording Input Modules from Libraries

To record a specific module from a library as an input to the current build step, call LIB$REC_DEPENDENCY twice. The first call specifies the VDE$K_DEPEND_INPUT_LIBRARY function code, as in the following example:

```
LIB$REC_DEPENDENCY( %REF(VDE$K_DEPEND_INPUT_LIBRARY),
                    %ASCID 'DEVD$:[DIR.SRCDIR]SYSMACLIB.MLB;1',
                    LIB_INDEX );
```

The **name-string** parameter is the full file specification of the library that contains the input module. The input module can be from an object library, a text library, or a macro library; in this example the module is from a macro library. The **lib-index** parameter is an output parameter that receives a *library index* that uniquely identifies the library. A library index is an integer value selected by LIB$REC_DEPENDENCY to represent the library.

The second call specifies the VDE$K_DEPEND_INPUT_MODULE function code, as in the following example:

```
LIB$REC_DEPENDENCY( %REF(VDE$K_DEPEND_INPUT_MODULE),
                    %ASCID 'INMODNAME',
                    LIB_INDEX );
```

This time the **name-string** parameter is the name of the input module you wish to specify. Thus, the **name-string** parameter is the name of an object module in an object library, a text element in a text library, or a macro in a macro library. The **lib-index** parameter is an input parameter that specifies the library index that uniquely identifies the library to which the module belongs. This is the library index returned by the previous VDE$K_DEPEND_INPUT_LIBRARY call. The library index thus establishes the connection between the library module (or element or macro) and the library to which it belongs.

If you record many input modules from the same library, record the library only once, using the VDE$K_DEPEND_INPUT_LIBRARY function code. Then make one additional call for each library module (or element or macro), specifying the same library index for each such call.

Recording Output Files

To record a specific file as an output from the current build step, call LIB$REC_DEPENDENCY with the VDE$K_DEPEND_OUTPUT_FILE function code, as in the following example:

```
LIB$REC_DEPENDENCY( %REF(VDE$K_DEPEND_OUTPUT_FILE),
                    %ASCID 'DEVD$:[DIR.OBJDIR]MOD1.LIS;3' );
```

The **name-string** parameter is the full file specification of the output file; this is the resultant name string that RMS returns. The **lib-index** parameter is not used.

Recording Output Modules for Libraries

To record a specific module in a library as an output from the current build step, call LIB$REC_DEPENDENCY twice. The first call specifies the VDE$K_DEPEND_OUTPUT_LIBRARY function code, as in the following example:

```
LIB$REC_DEPENDENCY( %REF(VDE$K_DEPEND_OUTPUT_LIBRARY),
                    %ASCID 'DEVD$:[DIR.OBJDIR]SYSOBJLIB.OLB;1',
                    LIB_INDEX );
```

The **name-string** parameter is the full file specification of the library that contains the output module. The output module can be from an object library, a text library, or a macro library; in this example the module is from an object library. The **lib-index** parameter is an output parameter that receives a *library index* that uniquely identifies this library. The library index is an integer value selected by LIB$REC_DEPENDENCY to represent the library.

The second call specifies the VDE$K_DEPEND_OUTPUT_MODULE function code, as in the following example:

```
LIB$REC_DEPENDENCY( %REF(VDE$K_DEPEND_OUTPUT_MODULE),
                    %ASCID 'OBJMODNAME',
                    LIB_INDEX );
```

This time the **name-string** parameter is the name of the output module you wish to specify. Thus, the **name-string** parameter is the name of an object module in an object library, a text element in a text library, or a macro in a macro library. The **lib-index** parameter is an input parameter that specifies the library index that uniquely identifies the library to which the module belongs. This is the library index returned by the previous VDE$K_DEPEND_OUTPUT_LIBRARY call. The library index thus establishes the connection between the library module (or element or macro) and the library to which it belongs.

If you record many modules as outputs from the same library, record the library only once, using the VDE$K_DEPEND_OUTPUT_LIBRARY function code. Then make one additional call for each library module, specifying the same library index for each such call.

Recording Object Modules

Object code that is used as input or output for a build step must be treated as a module in a library, even if the object code goes to or comes from an object file. In OpenVMS, an object file (such as MOD1.OBJ) contains one or more *object modules*. Each object module has its own module name; the object module is the unit that is copied to object libraries and is linked by the LINKER. OpenVMS stores dependency information for object code using object modules, not object files.

If an object file is an output of the build step, call LIB$REC_ DEPENDENCY using the VDE$K_DEPEND_OUTPUT_LIBRARY function code to specify the object file, as in the following example:

```
LIB$REC_DEPENDENCY( %REF(VDE$K_DEPEND_OUTPUT_LIBRARY),
                    %ASCID 'DEVD$:[DIR.OBJDIR]OBJFILE.OBJ;1',
                    LIB_INDEX );
```

In this call, the **name-string** parameter is the name of the object file. LIB$REC_DEPENDENCY returns a unique library index for the object file to the **lib-index** parameter.

Then call LIB$REC_DEPENDENCY using the VDE$K_DEPEND_ OUTPUT_MODULE function code for each object module in the object file, as in the following example:

```
LIB$REC_DEPENDENCY( %REF(VDE$K_DEPEND_OUTPUT_MODULE),
                    %ASCID 'OBJMODNAME',
                    LIB_INDEX );
```

To record object modules as inputs to a build step (as the LINKER must do), use the VDE$K_DEPEND_INPUT_LIBRARY and VDE$K_DEPEND_ INPUT_MODULE function codes in the same way.

End Recording Dependencies

To end the recording of inputs and outputs for the current build step, make a final call using the VDE$K_DEPEND_END_RECORD function code, as in the following example:

```
LIB$REC_DEPENDENCY( %REF(VDE$K_DEPEND_END_RECORD) );
```

This call tells LIB$REC_DEPENDENCY that there are no more inputs or outputs to record for the current build step. The dependency information passed in so far can be recorded in the OpenVMS database when the build step completes. The **name-string** and **lib-index** parameters are not used in this call.

Recording Is Done Only for Build Steps

If the language processor or other tool calls LIB$REC_DEPENDENCY when the processor or tool is not performing a build step, LIB$REC_ DEPENDENCY returns without performing any action. The VDE$K_ DEPEND_BEGIN_RECORD function checks whether a OpenVMS build step is being performed. If not, the function sets a flag that causes the remaining function codes to return immediately without spending any further time in LIB$REC_DEPENDENCY. LIB$REC_DEPENDENCY determines whether a build step is currently active by checking whether logical name VDE$RECORD_DEPENDENCY is defined to be ''TRUE'' and whether logical name VDE$PAGEFILE_SECTION is defined at all. It also requires the shareable image VDE$RECDEP to be present on the system. The overhead of calling LIB$REC_DEPENDENCY is thus designed to be negligible when the processor or tool is not performing a OpenVMS build step.

EXAMPLE

```
----- under construction -----
```

this section under construction...

C Callable Interface

This appendix describes the VDE callable interface.

Note: This interface is under development and is subject to change. Please contact VDE support *before* attempting to use this interface.

Table C–1 contains the itemlist codes used by various callback routines. Not all callbacks need nor use all codes listed in the table.

Table C–1 Itemlist Codes

Symbol	Description
VDE$_CI_END_OF_LIST	Signifies the end of the itemlist.
VDE$_CI_CHAIN	Reserved for future use.
VDE$_CI_ITEMLIST	Signifies the address of the itemlist. This entry allows the callback routine to locate the user-specified itemlist passed into the original call.
VDE$_CI_VERSION	Reserved for future use.
VDE$_CI_USER_CONTEXT	Ignored by VDE, available for any user purpose.
VDE$_CI_LIBRARY	Specifies the device and directory of the VDE library.
VDE$_CI_STREAM	Specifies the VDE stream.
VDE$_CI_FACILITY	Specifies the VDE facility.
VDE$_CI_MODULE	Specifies the VDE module.
VDE$_CI_REMARK	Specifies the remark assocated with the operation.
VDE$_CI_OUTPUT	Specifies the output device and directory specification. For operations such as fetches, this should contain only the device and directory; VDE will construct the file name and extension based on the module.
VDE$_CI_INPUT	Specifies the input device and directory. Do not specify a file name or extension.

Table C–1 (Cont.) Itemlist Codes

Symbol	Description
VDE$_CI_RESERVATION	Specifies the name of the reservation.
VDE$_CI_RESERVED	Received by the callback routine. A boolean flag that specifies if the module is currently reserved.
VDE$_CI_CAUSE	Specifies the cause for the replacement. The item references a longword containing the replacement cause code: VDE$K_CI_CAUSE_FOLD, VDE$K_CI_CAUSE_SCT, or VDE$K_CI_CAUSE_OTHER.
VDE$_CI_SCT_CONFERENCE	Specifies the SCT conference associated with the replacement.
VDE$_CI_SCT_NOTE	Specifies the SCT conference note associated with the replacement.
VDE$_CI_SCT_REMARK	Specifies the SCT remark associated with the replacement.
VDE$_CI_QAR_DATABASE	Specifies the QAR database associated with the replacement.
VDE$_CI_QAR	Specifies the QAR number associated with the replacement. Must be associated with a preceeding VDE$_CI_QAR_DATABASE item.
VDE$_CI_CLD	Specifies the CLD number associated with the replacement.

VDE$FetchModule—Fetches a Module

VDE$FetchModule fetches the specified module.

FORMAT

VDE$FetchModule *context, itemlist, callback, callbackparam, flags*

ARGUMENTS

context

type: **longword integer (signed)**
access: **read-write**
mechanism: **by reference**

This is the address of a longword used by VDE to reference a context structure. This longword is optional.

If this argument is specified, the first call must pass in the address of a longword zero. Once a VDE routine has been called, the contents of the address referenced by this longword must not be modified by the caller.

itemlist

type: **itemlist**
access: **read only**
mechanism: **by reference**

This is the address of an array of OpenVMS itemlist data structures.

```
struct VDEIL3
    {
    unsigned short int VDE$W_ITEMLENGTH;
    unsigned short int VDE$W_ITEMCODE;
    char *VDE$A_ITEMPOINTER;
    short int *VDE$A_ITEMRETURNLENGTH;
    };
```

The itemlist array must be terminated by a null-filled itemlist entry.

callback

type: **function**
access: **read only**
mechanism: **by reference**

The callback routine is called to process individual entries in a display operation. It is passed an itemlist containing information on each individual entry—the itemlist passed to the callback routine is distinct and seperate from the itemlist passed into the original request, though it does contain a pointer to the caller's original itemlist.

An example of the declaration of the callback routine follows:

```
int VDE$InterfaceTestShow( int, struct VDEIL3 *, int )
```

The first argument is not currently used. The second argument is the address of the itemlist. The third argument is the callback parameter.

The callback should return a successful status code.

callbackparam

type: **longword**
access: **read only**
mechanism: **by value**

The callback parameter is a longword value passed to the callback routine. It is not interpreted by VDE. The interpretation of the longword is left to the caller.

flags

type: **longword integer (signed)**
access: **read-only**
mechanism: **by value**

This is a bitflag and all unused bits should be specified as zero. The only bit currently defined is VDE$V_CI_FLAG_SIGNAL: this bit causes internal signals to generate text messages to SYS$OUTPUT: and SYS$ERROR:. If clear—the default—no messages will be displayed.

DESCRIPTION

VDE$FetchModule performs a fetch of the specified module. To perform a fetch, the library, stream, facility and module must be specified.

As parameters specified to VDE callback routines are ‘‘sticky’’, if a required parameter is not specified, it will be defaulted to the value used on a preceeding call.

EXAMPLE

```
struct VDEIL3 itmlst[MAX_ITEMLIST];
$DESCRIPTOR( sys$login, "SYS$LOGIN:" );
$DESCRIPTOR( library, "VDE$RESD2$:[VDE-MOTIF-DEMO.TSTLIB]" );
$DESCRIPTOR( stream, "MAIN" );
$DESCRIPTOR( facility, "MAIN" );
$DESCRIPTOR( module, "[MAIN]BLISS1.B32" );
n = 0;
itmlst[n].VDE$W_ItemCode                = VDE$_CI_LIBRARY;
itmlst[n].VDE$W_ItemLength              = library.dsc$w_length;
itmlst[n].VDE$A_ItemPointer             = library.dsc$a_pointer;
itmlst[n++].VDE$A_ItemReturnLength      = 0;

itmlst[n].VDE$W_ItemCode                = VDE$_CI_STREAM;
itmlst[n].VDE$W_ItemLength              = stream.dsc$w_length;
itmlst[n].VDE$A_ItemPointer             = stream.dsc$a_pointer;
itmlst[n++].VDE$A_ItemReturnLength      = 0;

itmlst[n].VDE$W_ItemCode                = VDE$_CI_FACILITY;
itmlst[n].VDE$W_ItemLength              = facility.dsc$w_length;
itmlst[n].VDE$A_ItemPointer             = facility.dsc$a_pointer;
itmlst[n++].VDE$A_ItemReturnLength      = 0;

itmlst[n].VDE$W_ItemCode                = VDE$_CI_MODULE;
itmlst[n].VDE$W_ItemLength              = module.dsc$w_length;
itmlst[n].VDE$A_ItemPointer             = module.dsc$a_pointer;
itmlst[n++].VDE$A_ItemReturnLength      = 0;
```

```
itmlst[n].VDE$W_ItemCode                = VDE$_CI_OUTPUT;
itmlst[n].VDE$W_ItemLength              = sys$login.dsc$w_length;
itmlst[n].VDE$A_ItemPointer             = sys$login.dsc$a_pointer;
itmlst[n++].VDE$A_ItemReturnLength      = 0;

itmlst[n].VDE$W_ItemCode                = 0;
itmlst[n].VDE$W_ItemLength              = 0;
itmlst[n].VDE$A_ItemPointer             = 0;
itmlst[n++].VDE$A_ItemReturnLength      = 0;

retstat = VDE$FetchModule( 0, itmlst, 0, 0 );
```

this section under construction...

VDE$ReplaceModule—Replaces a Module

VDE$ReplaceModule replaces the specified module.

FORMAT

VDE$ReplaceModule *context, itemlist, callback, callbackparam, flags*

ARGUMENTS

context

type: **longword integer (signed)**
access: **read-write**
mechanism: **by reference**

This is the address of a longword used by VDE to reference a context structure. This longword is optional.

If this argument is specified, the first call must pass in the address of a longword zero. Once a VDE routine has been called, the contents of the address referenced by this longword must not be modified by the caller.

itemlist

type: **itemlist**
access: **read only**
mechanism: **by reference**

This is the address of an array of OpenVMS itemlist data structures.

```
struct VDEIL3
    {
    unsigned short int VDE$W_ITEMLENGTH;
    unsigned short int VDE$W_ITEMCODE;
    char *VDE$A_ITEMPOINTER;
    short int *VDE$A_ITEMRETURNLENGTH;
    };
```

The itemlist array must be terminated by a null-filled itemlist entry.

See Table C–1 for a list of the itemlist codes.

callback

type: **function**
access: **read only**
mechanism: **by reference**

The callback routine is called to process individual entries in a display operation. It is passed an itemlist containing information on each individual entry—the itemlist passed to the callback routine is distinct and seperate from the itemlist passed into the original request, though it does contain a pointer to the caller's original itemlist.

An example of the declaration of the callback routine follows:

```
int VDE$InterfaceTestShow( int, struct VDEIL3 *, int )
```

The first argument is not currently used. The second argument is the address of the itemlist. The third argument is the callback parameter.

The callback should return a successful status code.

callbackparam

type: **longword**
access: **read only**
mechanism: **by value**

The callback parameter is a longword value passed to the callback routine. It is not interpreted by VDE. The interpretation of the longword is left to the caller.

flags

type: **longword integer (signed)**
access: **read-only**
mechanism: **by value**

This is a bitflag and all unused bits should be specified as zero. The only bit currently defined is VDE$V_CI_FLAG_SIGNAL: this bit causes internal signals to generate text messages to SYS$OUTPUT: and SYS$ERROR:. If clear—the default—no messages will be displayed.

DESCRIPTION

VDE$ReplaceModule replaces the specified module. To perform a replacement, the library, stream, facility and module must be specified. In addition, the replacement ‘‘cause’’ must be specified for certain VDE libraries.

As parameters specified to VDE callback routines are ‘‘sticky’’, if a required parameter is not specified, it will be defaulted to the value used on a preceeding call.

EXAMPLE

```
----- under construction -----
```

this section under construction...

VDE$SetFacility—Sets a Default Facility

VDE$SetFacility sets the specified facility as the default.

FORMAT

VDE$SetFacility *context, itemlist, callback, callbackparam, flags*

ARGUMENTS

context

type: **longword integer (signed)**
access: **read-write**
mechanism: **by reference**

This is the address of a longword used by VDE to reference a context structure. This longword is optional.

If this argument is specified, the first call must pass in the address of a longword zero. Once a VDE routine has been called, the contents of the address referenced by this longword must not be modified by the caller.

itemlist

type: **itemlist**
access: **read only**
mechanism: **by reference**

This is the address of an array of OpenVMS itemlist data structures.

```
struct VDEIL3
    {
    unsigned short int VDE$W_ITEMLENGTH;
    unsigned short int VDE$W_ITEMCODE;
    char *VDE$A_ITEMPOINTER;
    short int *VDE$A_ITEMRETURNLENGTH;
    };
```

The itemlist array must be terminated by a null-filled itemlist entry.

See Table C–1 for a list of the itemlist codes.

callback

type: **function**
access: **read only**
mechanism: **by reference**

The callback routine is called to process individual entries in a display operation. It is passed an itemlist containing information on each individual entry—the itemlist passed to the callback routine is distinct and seperate from the itemlist passed into the original request, though it does contain a pointer to the caller's original itemlist.

An example of the declaration of the callback routine follows:

```
int VDE$InterfaceTestShow( int, struct VDEIL3 *, int )
```

The first argument is not currently used. The second argument is the address of the itemlist. The third argument is the callback parameter.

The callback should return a successful status code.

callbackparam

type: **longword**
access: **read only**
mechanism: **by value**

The callback parameter is a longword value passed to the callback routine. It is not interpreted by VDE. The interpretation of the longword is left to the caller.

flags

type: **longword integer (signed)**
access: **read-only**
mechanism: **by value**

This is a bitflag and all unused bits should be specified as zero. The only bit currently defined is VDE$V_CI_FLAG_SIGNAL: this bit causes internal signals to generate text messages to SYS$OUTPUT: and SYS$ERROR:. If clear—the default—no messages will be displayed.

DESCRIPTION

VDE$SetFacility sets the specified facility as the default. Various commands and callbacks require this. Requires the specification of the the library, stream, and facility.

As parameters specified to VDE callback routines are ‘‘sticky’’, if a required parameter is not specified, it will be defaulted to the value used on a preceeding call.

EXAMPLE

```
int retstat;
int n;
struct VDEIL3 itmlst[20];
$DESCRIPTOR( sys$login, "SYS$LOGIN:" );
$DESCRIPTOR( library, "VDE$RESD2$:[VDE-MOTIF-DEMO.TSTLIB]" );
$DESCRIPTOR( stream, "MAIN" );
$DESCRIPTOR( facility, "MAIN" );
$DESCRIPTOR( module, "[MAIN]BLISS1.B32" );

n = 0;
itmlst[n].VDE$W_ItemCode                = VDE$_CI_LIBRARY;
itmlst[n].VDE$W_ItemLength              = library.dsc$w_length;
itmlst[n].VDE$A_ItemPointer             = library.dsc$a_pointer;
itmlst[n++].VDE$A_ItemReturnLength      = 0;

itmlst[n].VDE$W_ItemCode                = VDE$_CI_STREAM;
itmlst[n].VDE$W_ItemLength              = stream.dsc$w_length;
itmlst[n].VDE$A_ItemPointer             = stream.dsc$a_pointer;
itmlst[n++].VDE$A_ItemReturnLength      = 0;

itmlst[n].VDE$W_ItemCode                = VDE$_CI_FACILITY;
itmlst[n].VDE$W_ItemLength              = facility.dsc$w_length;
itmlst[n].VDE$A_ItemPointer             = facility.dsc$a_pointer;
itmlst[n++].VDE$A_ItemReturnLength      = 0;
```

```
itmlst[n].VDE$W_ItemCode                = 0;
itmlst[n].VDE$W_ItemLength              = 0;
itmlst[n].VDE$A_ItemPointer             = 0;
itmlst[n++].VDE$A_ItemReturnLength      = 0;

retstat = VDE$SetFacility( 0, itmlst, 0, 0 );
```

this section under construction...

VDE$SetLibrary—Opens a VDE Library

VDE$SetLibrary accesses the specified library.

FORMAT

VDE$SetLibrary *context, itemlist, callback, callbackparam, flags*

ARGUMENTS

context

type: **longword integer (signed)**
access: **read-write**
mechanism: **by reference**

This is the address of a longword used by VDE to reference a context structure. This longword is optional.

If this argument is specified, the first call must pass in the address of a longword zero. Once a VDE routine has been called, the contents of the address referenced by this longword must not be modified by the caller.

itemlist

type: **itemlist**
access: **read only**
mechanism: **by reference**

This is the address of an array of OpenVMS itemlist data structures.

```
struct VDEIL3
    {
    unsigned short int VDE$W_ITEMLENGTH;
    unsigned short int VDE$W_ITEMCODE;
    char *VDE$A_ITEMPOINTER;
    short int *VDE$A_ITEMRETURNLENGTH;
    };
```

The itemlist array must be terminated by a null-filled itemlist entry.

See Table C–1 for a list of the itemlist codes.

callback

type: **function**
access: **read only**
mechanism: **by reference**

The callback routine is called to process individual entries in a display operation. It is passed an itemlist containing information on each individual entry—the itemlist passed to the callback routine is distinct and seperate from the itemlist passed into the original request, though it does contain a pointer to the caller's original itemlist.

An example of the declaration of the callback routine follows:

```
int VDE$InterfaceTestShow( int, struct VDEIL3 *, int )
```

The first argument is not currently used. The second argument is the address of the itemlist. The third argument is the callback parameter.

The callback should return a successful status code.

callbackparam

type: **longword**
access: **read only**
mechanism: **by value**

The callback parameter is a longword value passed to the callback routine. It is not interpreted by VDE. The interpretation of the longword is left to the caller.

flags

type: **longword integer (signed)**
access: **read-only**
mechanism: **by value**

This is a bitflag and all unused bits should be specified as zero. The only bit currently defined is VDE$V_CI_FLAG_SIGNAL: this bit causes internal signals to generate text messages to SYS$OUTPUT: and SYS$ERROR:. If clear—the default—no messages will be displayed.

DESCRIPTION

VDE$SetLibrary accesses the specified library. This command can be explicitly called, or it can be implicitly called from within one of the other routines. To perform the access, the library must be specified.

As parameters specified to VDE callback routines are ‘‘sticky’’, if a required parameter is not specified, it will be defaulted to the value used on a preceeding call.

EXAMPLE

```
int retstat;
int n;
struct VDEIL3 itmlst[20];
$DESCRIPTOR( library, "VDE$RESD2$:[VDE-MOTIF-DEMO.TSTLIB]" );

n = 0;
itmlst[n].VDE$W_ItemCode                = VDE$_CI_LIBRARY;
itmlst[n].VDE$W_ItemLength              = library.dsc$w_length;
itmlst[n].VDE$A_ItemPointer             = library.dsc$a_pointer;
itmlst[n++].VDE$A_ItemReturnLength      = 0;

itmlst[n].VDE$W_ItemCode                = 0;
itmlst[n].VDE$W_ItemLength              = 0;
itmlst[n].VDE$A_ItemPointer             = 0;
itmlst[n++].VDE$A_ItemReturnLength      = 0;

retstat = VDE$SetLibrary( 0, itmlst, 0, 0 );
```

this section under construction...

VDE$SetStream—Sets the default stream

VDE$SetStream sets the default stream.

FORMAT

VDE$SetStream *context, itemlist, callback, callbackparam, flags*

ARGUMENTS

context

type: **longword integer (signed)**
access: **read-write**
mechanism: **by reference**

This is the address of a longword used by VDE to reference a context structure. This longword is optional.

If this argument is specified, the first call must pass in the address of a longword zero. Once a VDE routine has been called, the contents of the address referenced by this longword must not be modified by the caller.

itemlist

type: **itemlist**
access: **read only**
mechanism: **by reference**

This is the address of an array of OpenVMS itemlist data structures.

```
struct VDEIL3
    {
    unsigned short int VDE$W_ITEMLENGTH;
    unsigned short int VDE$W_ITEMCODE;
    char *VDE$A_ITEMPOINTER;
    short int *VDE$A_ITEMRETURNLENGTH;
    };
```

The itemlist array must be terminated by a null-filled itemlist entry.

See Table C–1 for a list of the itemlist codes.

callback

type: **function**
access: **read only**
mechanism: **by reference**

The callback routine is called to process individual entries in a display operation. It is passed an itemlist containing information on each individual entry—the itemlist passed to the callback routine is distinct and seperate from the itemlist passed into the original request, though it does contain a pointer to the caller's original itemlist.

An example of the declaration of the callback routine follows:

```
int VDE$InterfaceTestShow( int, struct VDEIL3 *, int )
```

The first argument is not currently used. The second argument is the address of the itemlist. The third argument is the callback parameter.

The callback should return a successful status code.

callbackparam

type: **longword**
access: **read only**
mechanism: **by value**

The callback parameter is a longword value passed to the callback routine. It is not interpreted by VDE. The interpretation of the longword is left to the caller.

flags

type: **longword integer (signed)**
access: **read-only**
mechanism: **by value**

This is a bitflag and all unused bits should be specified as zero. The only bit currently defined is VDE$V_CI_FLAG_SIGNAL: this bit causes internal signals to generate text messages to SYS$OUTPUT: and SYS$ERROR:. If clear—the default—no messages will be displayed.

DESCRIPTION

VDE$SetStream sets the default to the specified stream. This routine can be called explicitly, or implicitly from within another callback routine. To perform the operation, the library and stream must be specified.

As parameters specified to VDE callback routines are ‘‘sticky’’, if a required parameter is not specified, it will be defaulted to the value used on a preceeding call.

EXAMPLE

```
int retstat;
int n;
struct VDEIL3 itmlst[20];
$DESCRIPTOR( sys$login, "SYS$LOGIN:" );
$DESCRIPTOR( library, "VDE$RESD2$:[VDE-MOTIF-DEMO.TSTLIB]" );
$DESCRIPTOR( stream, "MAIN" );

n = 0;
itmlst[n].VDE$W_ItemCode                = VDE$_CI_LIBRARY;
itmlst[n].VDE$W_ItemLength              = library.dsc$w_length;
itmlst[n].VDE$A_ItemPointer             = library.dsc$a_pointer;
itmlst[n++].VDE$A_ItemReturnLength      = 0;

itmlst[n].VDE$W_ItemCode                = VDE$_CI_STREAM;
itmlst[n].VDE$W_ItemLength              = stream.dsc$w_length;
itmlst[n].VDE$A_ItemPointer             = stream.dsc$a_pointer;
itmlst[n++].VDE$A_ItemReturnLength      = 0;

itmlst[n].VDE$W_ItemCode                = 0;
itmlst[n].VDE$W_ItemLength              = 0;
itmlst[n].VDE$A_ItemPointer             = 0;
itmlst[n++].VDE$A_ItemReturnLength      = 0;

retstat = VDE$SetStream( 0, itmlst, 0, 0 );
```

this section under construction...

VDE$ShowFacilities—Displays facilities

VDE$ShowFacilities displays facilities within the specified stream.

FORMAT

VDE$ShowFacilities *context, itemlist, callback, callbackparam, flags*

ARGUMENTS

context

type: **longword integer (signed)**
access: **read-write**
mechanism: **by reference**

This is the address of a longword used by VDE to reference a context structure. This longword is optional.

If this argument is specified, the first call must pass in the address of a longword zero. Once a VDE routine has been called, the contents of the address referenced by this longword must not be modified by the caller.

itemlist

type: **itemlist**
access: **read only**
mechanism: **by reference**

This is the address of an array of OpenVMS itemlist data structures.

```
struct VDEIL3
    {
    unsigned short int VDE$W_ITEMLENGTH;
    unsigned short int VDE$W_ITEMCODE;
    char *VDE$A_ITEMPOINTER;
    short int *VDE$A_ITEMRETURNLENGTH;
    };
```

The itemlist array must be terminated by a null-filled itemlist entry.

See Table C–1 for a list of the itemlist codes.

callback

type: **function**
access: **read only**
mechanism: **by reference**

The callback routine is called to process individual entries in a display operation. It is passed an itemlist containing information on each individual entry—the itemlist passed to the callback routine is distinct and seperate from the itemlist passed into the original request, though it does contain a pointer to the caller's original itemlist.

An example of the declaration of the callback routine follows:

```
int VDE$InterfaceTestShow( int, struct VDEIL3 *, int )
```

The first argument is not currently used. The second argument is the address of the itemlist. The third argument is the callback parameter.

The callback should return a successful status code.

callbackparam

type: **longword**
access: **read only**
mechanism: **by value**

The callback parameter is a longword value passed to the callback routine. It is not interpreted by VDE. The interpretation of the longword is left to the caller.

flags

type: **longword integer (signed)**
access: **read-only**
mechanism: **by value**

This is a bitflag and all unused bits should be specified as zero. The only bit currently defined is VDE$V_CI_FLAG_SIGNAL: this bit causes internal signals to generate text messages to SYS$OUTPUT: and SYS$ERROR:. If clear—the default—no messages will be displayed.

DESCRIPTION

VDE$ShowFacilities displays the facilities within the specified library and stream. Each facility located results in a call to the callback routine. To perform the operation, the library and stream must be specified.

As parameters specified to VDE callback routines are ‘‘sticky’’, if a required parameter is not specified, it will be defaulted to the value used on a preceeding call.

EXAMPLE

```
int retstat;
int n;
struct VDEIL3 itmlst[20];
$DESCRIPTOR( library, "VDE$RESD2$:[VDE-MOTIF-DEMO.TSTLIB]" );
$DESCRIPTOR( stream, "MAIN" );
int VDE$InterfaceTestShow( int, struct VDEIL3 *, int );

n = 0;
itmlst[n].VDE$W_ItemCode                = VDE$_CI_LIBRARY;
itmlst[n].VDE$W_ItemLength              = library.dsc$w_length;
itmlst[n].VDE$A_ItemPointer             = library.dsc$a_pointer;
itmlst[n++].VDE$A_ItemReturnLength      = 0;

itmlst[n].VDE$W_ItemCode                = VDE$_CI_STREAM;
itmlst[n].VDE$W_ItemLength              = stream.dsc$w_length;
itmlst[n].VDE$A_ItemPointer             = stream.dsc$a_pointer;
itmlst[n++].VDE$A_ItemReturnLength      = 0;

itmlst[n].VDE$W_ItemCode                = 0;
itmlst[n].VDE$W_ItemLength              = 0;
itmlst[n].VDE$A_ItemPointer             = 0;
itmlst[n++].VDE$A_ItemReturnLength      = 0;

retstat = VDE$ShowFacilities( 0, itmlst, VDE$InterfaceTestShow, 0x0048B );
```

this section under construction...

VDE$ShowProjects—Displays projects

VDE$ShowProjects displays projects within the specified stream.

FORMAT

VDE$ShowProjects *context, itemlist, callback, callbackparam, flags*

ARGUMENTS

context

type: **longword integer (signed)**
access: **read-write**
mechanism: **by reference**

This is the address of a longword used by VDE to reference a context structure. This longword is optional.

If this argument is specified, the first call must pass in the address of a longword zero. Once a VDE routine has been called, the contents of the address referenced by this longword must not be modified by the caller.

itemlist

type: **itemlist**
access: **read only**
mechanism: **by reference**

This is the address of an array of OpenVMS itemlist data structures.

```
struct VDEIL3
    {
    unsigned short int VDE$W_ITEMLENGTH;
    unsigned short int VDE$W_ITEMCODE;
    char *VDE$A_ITEMPOINTER;
    short int *VDE$A_ITEMRETURNLENGTH;
    };
```

The itemlist array must be terminated by a null-filled itemlist entry.

See Table C–1 for a list of the itemlist codes.

callback

type: **function**
access: **read only**
mechanism: **by reference**

The callback routine is called to process individual entries in a display operation. It is passed an itemlist containing information on each individual entry—the itemlist passed to the callback routine is distinct and seperate from the itemlist passed into the original request, though it does contain a pointer to the caller's original itemlist.

An example of the declaration of the callback routine follows:

```
int VDE$InterfaceTestShow( int, struct VDEIL3 *, int )
```

The first argument is not currently used. The second argument is the address of the itemlist. The third argument is the callback parameter.

The callback should return a successful status code.

callbackparam

type: **longword**
access: **read only**
mechanism: **by value**

The callback parameter is a longword value passed to the callback routine. It is not interpreted by VDE. The interpretation of the longword is left to the caller.

flags

type: **longword integer (signed)**
access: **read-only**
mechanism: **by value**

This is a bitflag and all unused bits should be specified as zero. The only bit currently defined is VDE$V_CI_FLAG_SIGNAL: this bit causes internal signals to generate text messages to SYS$OUTPUT: and SYS$ERROR:. If clear—the default—no messages will be displayed.

DESCRIPTION

VDE$ShowProjects displays the projects within the specified library and stream. Each facility located results in a call to the callback routine. To perform the operation, the library and stream must be specified.

As parameters specified to VDE callback routines are ‘‘sticky’’, if a required parameter is not specified, it will be defaulted to the value used on a preceeding call.

EXAMPLE

```
   int RetStat;
   int n;
   struct vdeil3 itmlst[MAX_IL3];
   char *test_name, *test_remark;
   int test_length1, test_length2;
   int NullArg = 0;

       n = 0;
       test_name = "Other Project";
       test_length1 = strlen (test_name);
itmlst[n].vde$w_itemlength  = (unsigned short) (unsigned char) test_length1;
itmlst[n].vde$w_itemcode  = VDE$_CI_PROJECT;
itmlst[n].vde$a_itempointer  = (char *) test_name;
itmlst[n++].vde$a_itemreturnlength = 0;

test_remark = "this remark is for the Other Project";
test_length2 = strlen (test_remark);
itmlst[n].vde$w_itemlength  = (unsigned short) (unsigned char) test_length2;
itmlst[n].vde$w_itemcode  = VDE$_CI_REMARK;
itmlst[n].vde$a_itempointer  = (char *) test_remark;
itmlst[n++].vde$a_itemreturnlength = 0;
```

```
itmlst[n].vde$w_itemlength  = 0;
itmlst[n].vde$w_itemcode  = 0;
itmlst[n].vde$a_itempointer  = 0;
itmlst[n++].vde$a_itemreturnlength = 0;

   retstat = VDE$ShowProjects( 0, itmlst, VDE$InterfaceTestShow, 0x0048B );
```

this section under construction...

VDE$ShowLibrary—Displays information about the library

VDE$ShowLibrary displays information about the specified library.

FORMAT

VDE$ShowLibrary *context, itemlist, callback, callbackparam, flags*

ARGUMENTS

context

type: **longword integer (signed)**
access: **read-write**
mechanism: **by reference**

This is the address of a longword used by VDE to reference a context structure. This longword is optional.

If this argument is specified, the first call must pass in the address of a longword zero. Once a VDE routine has been called, the contents of the address referenced by this longword must not be modified by the caller.

itemlist

type: **itemlist**
access: **read only**
mechanism: **by reference**

This is the address of an array of OpenVMS itemlist data structures.

```
struct VDEIL3
    {
    unsigned short int VDE$W_ITEMLENGTH;
    unsigned short int VDE$W_ITEMCODE;
    char *VDE$A_ITEMPOINTER;
    short int *VDE$A_ITEMRETURNLENGTH;
    };
```

The itemlist array must be terminated by a null-filled itemlist entry.

See Table C–1 for a list of the itemlist codes.

callback

type: **function**
access: **read only**
mechanism: **by reference**

The callback routine is called to process individual entries in a display operation. It is passed an itemlist containing information on each individual entry—the itemlist passed to the callback routine is distinct and seperate from the itemlist passed into the original request, though it does contain a pointer to the caller's original itemlist.

An example of the declaration of the callback routine follows:

```
int VDE$InterfaceTestShow( int, struct VDEIL3 *, int )
```

The first argument is not currently used. The second argument is the address of the itemlist. The third argument is the callback parameter.

The callback should return a successful status code.

callbackparam

type: **longword**
access: **read only**
mechanism: **by value**

The callback parameter is a longword value passed to the callback routine. It is not interpreted by VDE. The interpretation of the longword is left to the caller.

flags

type: **longword integer (signed)**
access: **read-only**
mechanism: **by value**

This is a bitflag and all unused bits should be specified as zero. The only bit currently defined is VDE$V_CI_FLAG_SIGNAL: this bit causes internal signals to generate text messages to SYS$OUTPUT: and SYS$ERROR:. If clear—the default—no messages will be displayed.

DESCRIPTION

VDE$ShowLibrary displays information about the specified library. Information is returned via a call to the callback routine. To perform the operation, the library must be specified.

As parameters specified to VDE callback routines are ‘‘sticky’’, if a required parameter is not specified, it will be defaulted to the value used on a preceeding call.

EXAMPLE

```
int retstat;
int n;
struct VDEIL3 itmlst[20];
$DESCRIPTOR( library, "VDE$RESD2$:[VDE-MOTIF-DEMO.TSTLIB]" );
int VDE$InterfaceTestShow( int, struct VDEIL3 *, int );

n = 0;
itmlst[n].VDE$W_ItemCode                = VDE$_CI_LIBRARY;
itmlst[n].VDE$W_ItemLength              = library.dsc$w_length;
itmlst[n].VDE$A_ItemPointer             = library.dsc$a_pointer;
itmlst[n++].VDE$A_ItemReturnLength      = 0;

itmlst[n].VDE$W_ItemCode                = 0;
itmlst[n].VDE$W_ItemLength              = 0;
itmlst[n].VDE$A_ItemPointer             = 0;
itmlst[n++].VDE$A_ItemReturnLength      = 0;

retstat = VDE$ShowLibrary( 0, itmlst, VDE$InterfaceTestShow, 0x0048B  );
```

this section under construction...

VDE$ShowModules—Displays modules

VDE$ShowModules displays modules within a facility.

FORMAT

VDE$ShowModules *context, itemlist, callback, callbackparam, flags*

ARGUMENTS

context

type: **longword integer (signed)**
access: **read-write**
mechanism: **by reference**

This is the address of a longword used by VDE to reference a context structure. This longword is optional.

If this argument is specified, the first call must pass in the address of a longword zero. Once a VDE routine has been called, the contents of the address referenced by this longword must not be modified by the caller.

itemlist

type: **itemlist**
access: **read only**
mechanism: **by reference**

This is the address of an array of OpenVMS itemlist data structures.

```
struct VDEIL3
    {
    unsigned short int VDE$W_ITEMLENGTH;
    unsigned short int VDE$W_ITEMCODE;
    char *VDE$A_ITEMPOINTER;
    short int *VDE$A_ITEMRETURNLENGTH;
    };
```

The itemlist array must be terminated by a null-filled itemlist entry.

See Table C–1 for a list of the itemlist codes.

callback

type: **function**
access: **read only**
mechanism: **by reference**

The callback routine is called to process individual entries in a display operation. It is passed an itemlist containing information on each individual entry—the itemlist passed to the callback routine is distinct and seperate from the itemlist passed into the original request, though it does contain a pointer to the caller's original itemlist.

An example of the declaration of the callback routine follows:

```
int VDE$InterfaceTestShow( int, struct VDEIL3 *, int )
```

The first argument is not currently used. The second argument is the address of the itemlist. The third argument is the callback parameter.

The callback should return a successful status code.

callbackparam

type: **longword**
access: **read only**
mechanism: **by value**

The callback parameter is a longword value passed to the callback routine. It is not interpreted by VDE. The interpretation of the longword is left to the caller.

flags

type: **longword integer (signed)**
access: **read-only**
mechanism: **by value**

This is a bitflag and all unused bits should be specified as zero. The only bit currently defined is VDE$V_CI_FLAG_SIGNAL: this bit causes internal signals to generate text messages to SYS$OUTPUT: and SYS$ERROR:. If clear—the default—no messages will be displayed.

DESCRIPTION

VDE$ShowModules displays all modules within a specified facility. Each module located results in a call to the callback routine. To perform the operation, the library, stream and facility module must be specified.

As parameters specified to VDE callback routines are ‘‘sticky’’, if a required parameter is not specified, it will be defaulted to the value used on a preceeding call.

EXAMPLE

```
int retstat;
int n;
struct VDEIL3 itmlst[20];
$DESCRIPTOR( library, "VDE$RESD2$:[VDE-MOTIF-DEMO.TSTLIB]" );
$DESCRIPTOR( stream, "MAIN" );
$DESCRIPTOR( facility, "MAIN" );
int VDE$InterfaceTestShow( int, struct VDEIL3 *, int );

n = 0;
itmlst[n].VDE$W_ItemCode                = VDE$_CI_LIBRARY;
itmlst[n].VDE$W_ItemLength              = library.dsc$w_length;
itmlst[n].VDE$A_ItemPointer             = library.dsc$a_pointer;
itmlst[n++].VDE$A_ItemReturnLength      = 0;

itmlst[n].VDE$W_ItemCode                = VDE$_CI_STREAM;
itmlst[n].VDE$W_ItemLength              = stream.dsc$w_length;
itmlst[n].VDE$A_ItemPointer             = stream.dsc$a_pointer;
itmlst[n++].VDE$A_ItemReturnLength      = 0;

itmlst[n].VDE$W_ItemCode                = VDE$_CI_FACILITY;
itmlst[n].VDE$W_ItemLength              = facility.dsc$w_length;
itmlst[n].VDE$A_ItemPointer             = facility.dsc$a_pointer;
itmlst[n++].VDE$A_ItemReturnLength      = 0;
```

```
itmlst[n].VDE$W_ItemCode                = 0;
itmlst[n].VDE$W_ItemLength              = 0;
itmlst[n].VDE$A_ItemPointer             = 0;
itmlst[n++].VDE$A_ItemReturnLength      = 0;

retstat = VDE$ShowModules( 0, itmlst, VDE$InterfaceTestShow, 0x0048B );
```

this section under construction...

VDE$ShowStreams—Displays streams

VDE$ShowStreams fetches the specified module.

FORMAT

VDE$ShowStreams *context, itemlist, callback, callbackparam, flags*

ARGUMENTS

context

type: **longword integer (signed)**
access: **read-write**
mechanism: **by reference**

This is the address of a longword used by VDE to reference a context structure. This longword is optional.

If this argument is specified, the first call must pass in the address of a longword zero. Once a VDE routine has been called, the contents of the address referenced by this longword must not be modified by the caller.

itemlist

type: **itemlist**
access: **read only**
mechanism: **by reference**

This is the address of an array of OpenVMS itemlist data structures.

```
struct VDEIL3
    {
    unsigned short int VDE$W_ITEMLENGTH;
    unsigned short int VDE$W_ITEMCODE;
    char *VDE$A_ITEMPOINTER;
    short int *VDE$A_ITEMRETURNLENGTH;
    };
```

The itemlist array must be terminated by a null-filled itemlist entry.

See Table C–1 for a list of the itemlist codes.

callback

type: **function**
access: **read only**
mechanism: **by reference**

The callback routine is called to process individual entries in a display operation. It is passed an itemlist containing information on each individual entry—the itemlist passed to the callback routine is distinct and seperate from the itemlist passed into the original request, though it does contain a pointer to the caller's original itemlist.

An example of the declaration of the callback routine follows:

```
int VDE$InterfaceTestShow( int, struct VDEIL3 *, int )
```

The first argument is not currently used. The second argument is the address of the itemlist. The third argument is the callback parameter.

The callback should return a successful status code.

callbackparam

type: **longword**
access: **read only**
mechanism: **by value**

The callback parameter is a longword value passed to the callback routine. It is not interpreted by VDE. The interpretation of the longword is left to the caller.

flags

type: **longword integer (signed)**
access: **read-only**
mechanism: **by value**

This is a bitflag and all unused bits should be specified as zero. The only bit currently defined is VDE$V_CI_FLAG_SIGNAL: this bit causes internal signals to generate text messages to SYS$OUTPUT: and SYS$ERROR:. If clear—the default—no messages will be displayed.

DESCRIPTION

VDE$ShowStreams displays the streams present in the specified library. Each stream located results in a call to the callback routine. To perform the operation, the library must be specified.

As parameters specified to VDE callback routines are ‘‘sticky’’, if a required parameter is not specified, it will be defaulted to the value used on a preceeding call.

EXAMPLE

```
int retstat;
int n;
struct VDEIL3 itmlst[MAX_ITEMLIST];
$DESCRIPTOR( sys$login, "SYS$LOGIN:" );
$DESCRIPTOR( library, "VDE$RESD2$:[VDE-MOTIF-DEMO.TSTLIB]" );
$DESCRIPTOR( stream, "MAIN" );
$DESCRIPTOR( facility, "MAIN" );
$DESCRIPTOR( module, "[MAIN]BLISS1.B32" );
int VDE$InterfaceTestShow( int, struct VDEIL3 *, int );

n = 0;
itmlst[n].VDE$W_ItemCode                = VDE$_CI_LIBRARY;
itmlst[n].VDE$W_ItemLength              = library.dsc$w_length;
itmlst[n].VDE$A_ItemPointer             = library.dsc$a_pointer;
itmlst[n++].VDE$A_ItemReturnLength      = 0;

itmlst[n].VDE$W_ItemCode                = VDE$_CI_STREAM;
itmlst[n].VDE$W_ItemLength              = stream.dsc$w_length;
itmlst[n].VDE$A_ItemPointer             = stream.dsc$a_pointer;
itmlst[n++].VDE$A_ItemReturnLength      = 0;

itmlst[n].VDE$W_ItemCode                = VDE$_CI_FACILITY;
itmlst[n].VDE$W_ItemLength              = facility.dsc$w_length;
itmlst[n].VDE$A_ItemPointer             = facility.dsc$a_pointer;
itmlst[n++].VDE$A_ItemReturnLength      = 0;
```

```
itmlst[n].VDE$W_ItemCode                = 0;
itmlst[n].VDE$W_ItemLength              = 0;
itmlst[n].VDE$A_ItemPointer             = 0;
itmlst[n++].VDE$A_ItemReturnLength      = 0;

retstat = VDE$ShowStreams( 0, itmlst, VDE$InterfaceTestShow, 0x00246 );
```

this section under construction...

VDE$UnreserveModule—Unreserve a Module

VDE$UnreserveModule cancels a previous module reservation.

FORMAT

VDE$UnreserveModule *context, itemlist, callback, callbackparam, flags*

ARGUMENTS

context

type: **longword integer (signed)**
access: **read-write**
mechanism: **by reference**

This is the address of a longword used by VDE to reference a context structure. This longword is optional.

If this argument is specified, the first call must pass in the address of a longword zero. Once a VDE routine has been called, the contents of the address referenced by this longword must not be modified by the caller.

itemlist

type: **itemlist**
access: **read only**
mechanism: **by reference**

This is the address of an array of OpenVMS itemlist data structures.

```
struct VDEIL3
    {
    unsigned short int VDE$W_ITEMLENGTH;
    unsigned short int VDE$W_ITEMCODE;
    char *VDE$A_ITEMPOINTER;
    short int *VDE$A_ITEMRETURNLENGTH;
    };
```

The itemlist array must be terminated by a null-filled itemlist entry.

See Table C–1 for a list of the itemlist codes.

callback

type: **function**
access: **read only**
mechanism: **by reference**

The callback routine is called to process individual entries in a display operation. It is passed an itemlist containing information on each individual entry—the itemlist passed to the callback routine is distinct and seperate from the itemlist passed into the original request, though it does contain a pointer to the caller's original itemlist.

An example of the declaration of the callback routine follows:

```
int VDE$InterfaceTestShow( int, struct VDEIL3 *, int )
```

The first argument is not currently used. The second argument is the address of the itemlist. The third argument is the callback parameter.

The callback should return a successful status code.

callbackparam

type: **longword**
access: **read only**
mechanism: **by value**

The callback parameter is a longword value passed to the callback routine. It is not interpreted by VDE. The interpretation of the longword is left to the caller.

flags

type: **longword integer (signed)**
access: **read-only**
mechanism: **by value**

This is a bitflag and all unused bits should be specified as zero. The only bit currently defined is VDE$V_CI_FLAG_SIGNAL: this bit causes internal signals to generate text messages to SYS$OUTPUT: and SYS$ERROR:. If clear—the default—no messages will be displayed.

DESCRIPTION

VDE$UnreserveModule cancels a previous reservation on the specified module. To perform the operation, the library, stream, facility and module must be specified.

As parameters specified to VDE callback routines are ‘‘sticky’’, if a required parameter is not specified, it will be defaulted to the value used on a preceeding call.

EXAMPLE

```
----- under construction -----
```

this section under construction...

D CMS Library Uploader

The OpenVMS Development Environment (VDE) tool VDE$UPLOAD assist OpenVMS engineers in uploading the contents of multiple CMS libraries into VDE.

VDE$UPLOAD requires information on environment to be created and the CMS information to be uploaded, and the location(s) where the VDE database is to be created. VDE$UPLOAD will then create a new VDE database based on the specified CMS libraries. As part of this, specified CMS classes from the specified CMS libraries will form the basis of new VDE streams, the users from the CMS libraries will be created in the VDE database, and the CMS command notification ACLs will be uploaded into VDE.

VDE$UPLOAD does not alter the CMS libraries in any way. As a consequence, the manual deletion of CMS command notification ACLs may be required.

To use VDE$UPLOAD, you must create a control file describing the requested operation. See Table D–1 for the syntax, and see the attached example file.

Table D–1 VDE$UPLOAD Syntax

Library Block	
	The library block contains general information on the library, including both descriptive and location information. The library block begins a new library creation operation, and subsequent literal and stream blocks apply to the VDE database created by the immediately previous library block. Multiple creation operations can be accomplished in a single data file.
BEGIN LIBRARY	
END LIBRARY	
	Library block delimiters. Cause the creation of a VDE library, using the other parameters in this section of the table.
MNEMONIC FOOBAR	
	The mnemonic name to be associated with the new library. This name must meet VDE naming requirements.
CMS FOOBAR_CMS_LIB_ROOT$:[000000]	
	The full specification of the CMS library (libraries) to be uploaded. This directory contains one or more CMS library subdirectories, each of these subdirectories with a [.CMS] subdirectory containing a CMS library.
IDENTIFIER VMS_IDENTIFIER_NAME	
	The name of an OpenVMS security identifier required for access to the created VDE libary.
ROOT 11DUA113:[VDE.FOOBAR.DBROOT]	

Table D–1 (Cont.) VDE$UPLOAD Syntax

Library Block
The root directory specification for the new database. This is the area that contains the VDE$DATABASE.RDB file.
DATA 11DUA113:[VDE.FOOBAR.DBDATA]
The data directory specification for the new database. This is the area that contains the VDE$DATABASE.RDA file.
SNAPSHOT 11DUA113:[VDE.FOOBAR.DBSNP]
The snapshot directory specification for the new database. This is the area that contains the VDE$DATABASE.SNP file.
JOURNAL 11DUA113:[VDE.FOOBAR.DBJNL]
The journal directory specification for the new database. This is the area that contains the VDE$DATABASE.JNL file.
STAGING 11DUA113:[VDE.FOOBAR.VDE$STAGING]
The staging directory specification for the new database. This area will contain the staging area subdirectories.
BUILDING 11DUA113:[VDE.FOOBAR.VDE$BUILDING]
The building directory specification for the new database. This area will contain the building area subdirectories.
Stream Block
The stream block contains stream-related information, including the specific CMS classes that will be uploaded into VDE streams.
BEGIN STREAM
END STREAM
Stream block delimiters. Cause the creation of a VDE stream, using the other parameters in this section of the table.
NAME V6.1_SSB
The name of the new stream. Must meet VDE naming requirements. Must also match the name of a CMS class that is present in the specified CMS libraries.
MAINLINE TRUE
Indicates the stream is (TRUE) or is not (FALSE) a member of the main line of descent in the VDE database.
PARENT MAIN
The name of the parent of the new stream, or MAIN for the first stream in the database.
SUCCESSORS V6.2_SSB
The name of a successor stream, a comma-seperated list of successor streams, or the keyword NONE if there are no successors to this stream.
CONFERENCE V61-YELLOW
The name of the notes conference used internally by VDE to track all activity for this stream. This conference is *not* the SCT (Source Code Tracking) notes conference.

Table D–1 (Cont.) VDE$UPLOAD Syntax

Literal Block	
	The literal block contains VDE commands that are to be directly executed as part of the database creation operation.
BEGIN LITERAL	
END LITERAL	
	Literal block delimiters. This block causes the procedure to emit the specified VDE commands that were specified in this block after the creation of the VDE library and the associated streams. Most often used to update replacement policy and pre-created users and user privilege profiles.

```
!
! Example VDE$UPLOAD.DAT control file
!
BEGIN LIBRARY
MNEMONIC FOOBAR
CMS FOOBAR_CMS_LIB_ROOT$:[000000]
IDENTIFIER VMS_IDENTIFIER_NAME
ROOT $11$DUA113:[VDE.FOOBAR.DBROOT]
DATA $11$DUA113:[VDE.FOOBAR.DBDATA]
SNAPSHOT $11$DUA113:[VDE.FOOBAR.DBSNP]
JOURNAL $11$DUA113:[VDE.FOOBAR.DBJNL]
STAGING $11$DUA113:[VDE.FOOBAR.VDE$STAGING]
BUILDING $11$DUA113:[VDE.FOOBAR.VDE$BUILDING]
END LIBRARY

BEGIN STREAM
NAME V6.1_SSB
MAINLINE TRUE
PARENT MAIN
SUCCESSORS V6.2_SSB
CONFERENCE V61-YELLOW
END STREAM

BEGIN STREAM
NAME V6.2_SSB
MAINLINE TRUE
PARENT V6.1_SSB
SUCCESSORS NONE
CONFERENCE V62-YELLOW
END STREAM

BEGIN LITERAL
CREATE USER SYSTEM    /NAME="System Manager"/DEF_P=ALL/AUTH_P=ALL/REMARK="System De
Modify Stream V6.1_SSB/replacement=immediate/notify=marshall /Remark="VAX V6.1 Stre
Modify Stream V6.2_SSB/replacement=immediate/notify=marshall /Remark="VAX V6.2 Stre
END LITERAL
!
! End VDE$UPLOAD.DAT
!
```

E Error Messages

This appendix describes the error messages that can be reported by VDE.

ABKEYW, ambiguous qualifier or keyword '*text*' - supply more characters

Facility: VDE, OpenVMS Development Environment

Explanation: VDE cannot complete the command because '*text*' is ambiguous in the context of this command.

ABTBADST, VDE may be in an inconsistent internal state

Facility: VDE, OpenVMS Development Environment

Explanation: The user has typed 3 or more CTRL/Cs. VDE normally responds to a CTRL/C by continuing execution until a consistent internal state is achieved and then aborts the current command. However, if VDE is in an infinite loop and does not detect this "consistent" state, it will abort anyway after the 3rd CTRL/C.

User Action: Leave and enter VDE again to make sure the internal state is consistent.

ABTCNTLC, command aborted by Control-C

Facility: VDE, OpenVMS Development Environment

Explanation: The current VDE command has been aborted due to a control C issued by the user. VDE will attempt to gracefully recover.

User Action: Issue another command to VDE. It has gracefully to a consistent internal state.

ADATETIME, accepted.checkin.datetime: *datetime*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

AERRORS, accepted.checkin.defectsfixed: *defects*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

AGROUP, accepted.checkin.group: *group*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

AITEM, accepted.checkin.item: *item*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

AMODULE, accepted.checkin.module: *modname*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

ANOTE, accepted.checkin.note: *topic.reply*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

ARCHADDED, architecture *architecture* added to the database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated architecture definition has been added to the VDE database. This message is only issued when you use the /LOG qualifier on the CREATE ARCHITECTURE command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

ARCHDEL, architecture *architecture* deleted from the database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the definition for the indicated architecture has been deleted from the VDE database. This

message is only issued when you use the /LOG qualifier on the DELETE ARCHITECTURE command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

ARCHEXISTS, architecture *architecture* already exists in database; not changed

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated architecture name was specified on a CREATE ARCHITECTURE command, but an entry for that architecture is already present in the database. Hence a new entry was not created for that architecture. Alternatively, the indicated name was specified as the new name for an existing architecture on the MODIFY ARCHITECTURE command, but another architecture already has the requested new name. Hence the name of the existing architecture could not be changed to the new name.

User Action: If you entered the wrong architecture name, reenter the command with the correct name. If you entered the right architecture name, use the SHOW ARCHITECTURE command to display the database information for that architecture. If you are not satisfied with that information, use the MODIFY ARCHITECTURE command to change it.

ARCHMOD, information for architecture *architecture* modified in the database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that information for the indicated architecture has been modified in the VDE database. This message is only issued when you use the /LOG qualifier on the MODIFY ARCHITECTURE command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

ARCHNOTDEL, architecture *architecture* not deleted

Facility: VDE, OpenVMS Development Environment

Explanation: You are attempting to delete an architecture that cannot be deleted. Additional messages are normally included with this message, and these messages indicate why the architecture cannot be deleted.

User Action: Resolve the problem(s) flagged by the other error messages, and reenter the DELETE ARCHITECTURE command.

ARCHNOTFOUND, architecture *architecture* not found in database; parameter ignored

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated architecture name was entered on a SHOW ARCHITECTURE command, but no such architecture was found in the VDE database.

User Action: This is not necessarily an error; the message simply tells you that the indicated architecture does not exist in the database.

ARCHSYNTAX, invalid architecture name syntax: *name*

Facility: VDE, OpenVMS Development Environment

Explanation: The name or format specified for the architecture name is not using valid syntax.

User Action: Respecify the architecture name using only alphanumeric, underscore, dollar sign and hyphen characters.

AREASON, accepted.checkin.reasonmask: *reason*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

AREPLID, accepted.checkin.replacementid: *replid*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

AREPLNAME, accepted.checkin.replname: *replname*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

ARMPBCMSDIR, accepted.module.cmslibrary: *cmsdir*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

ARMPBCOUNT, accepted.module.count: *count*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

ARMPBGENNEW, accepted.module.newgeneration: *newgen*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

ARMPBGENOLD, accepted.module.oldgeneration: *oldgen*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

ARMPBMODFLAG, accepted.module.modflags: *modflags*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

ARMPBMODNAME, accepted.module.modname: *modname*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

ARMPBREPLREMARK, accepted.module.replaceremark: *replaceremark*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

ARMPBRESREMARK, accepted.module.reserveremark: *reserveremark*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

ARMPBSTAGING, accepted.module.staging: *staging*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message. ! ========== ========== ========== ==========

ASTREAM, accepted.checkin.stream: *stream*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

ATTACHED, terminal now attached to process *process-name*

Facility: VDE, OpenVMS Development Environment

Explanation: The SPAWN command issues this message before attaching to the new process.

ATYPCOD, accepted.checkin.typecode: *type*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

ATYPNAM, accepted.checkin.typename: *type*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

AUSERNAME, accepted.checkin.username: *username*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message. ! ==========

BADASSUMPTION, bad assumption, *name*, *value*, *value*

Facility: VDE, OpenVMS Development Environment

Explanation: VDE has detected an inconsistency in its internal memory pool.

User Action: Please contact VDE support for assistance. Please include any information that may be useful or relevent in duplicating the problem.

BADCHAR, illegal character encountered

Facility: VDE, OpenVMS Development Environment

Explanation: An unanticipated character was found on the command line. This character may be found to have meaning in other syntaxes; however, it lacks meaning in this context.

User Action: Verify the appropriate syntax of the specific command.

BADFILNAM, illegal file specification '*text*'

Facility: VDE, OpenVMS Development Environment

Explanation: The provided file name does not follow standard VMS file specification conventions.

BADIMAFLA, bad image flavor detected

Facility: VDE, OpenVMS Development Environment

Explanation: VDE has detected an inconsistent image flavor setting.

User Action: Please contact VDE support for assistance. Please include any information that may be useful or relevent in duplicating the problem.

BADMARKER, bad marker file format for generation *generation*

Facility: VDE, OpenVMS Development Environment

Explanation: The CMS element for a module should contain a marker file that points to the actual source module, but the indicated generation of the module does not contain a valid marker file. This error is also seen when /OUTPUT=NLA0: is specified on a FETCH or RESERVE of a module that uses a marker file.

User Action: Examine the text of the CMS generation using CMS directly to determine what is wrong. If the module should not use marker files, change its marker-file attribute with the MODIFY MODULE command. If it should use marker files, determine what is wrong with the CMS element and repair it with CMS so it points to the appropriate actual

source module. If /OUTPUT=NLA0: was specified, respecify the command with a non-null device and directory.

BADQUESTAT, bad status for batch queue *queue-name*

Facility: VDE, OpenVMS Development Environment

Explanation: A batch queue to which you are attempting to submit a build job is not available. The reason why it is not available is given by a second message following this message.

User Action: This is an informational message only. If you need the displayed batch queue for your build job, stop the build job and resubmit it later when the queue is available again.

BADSCRATCH, bad SYS$SCRATCH directory

Facility: VDE, OpenVMS Development Environment

Explanation: The directory specification associated with the SYS$SCRATCH logical name is not syntactically correct. Associated with the VDE$_INVDIRSPEC error.

User Action: Reenter the VDE command after redefining SYS$SCRATCH to a valid directory specification.

BADSTATUS, bad status returned from *routine-name*

Facility: VDE, OpenVMS Development Environment

Explanation: VDE got an unexpected error status from a system service or RTL routine *routine-name* that it called.

User Action: Examine the associated error messages and determine if the problem can be remedied through changes to process quotas or to SYSGEN parameters, or due to a site-specific configuration problem. Contact VDE support for assistance.

BADVMSVERS, unable to interpret OpenVMS version string: *VMS-version*

Facility: VDE, OpenVMS Development Environment

Explanation: VDE is unable to determine OpenVMS version information from the version string returned either from a SYS$GETSYI service call or from a translation of the logical name VDE$VMS_VERSION. This may indicate you are running an internal baselevel of the operating system.

User Action: Define the logical name VDE$VMS_VERSION to a valid OpenVMS version string value. This will override the value returned by SYS$GETSYI.

BINSRCMOD, binary source modules cannot be displayed or differenced

Facility: VDE, OpenVMS Development Environment

Explanation: You have entered a SHOW or EDIT command to display a binary (non-text) source module or the differences between two generations

of a binary source module. These commands do not work on binary modules.

User Action: Do not use the SHOW or EDIT command on binary source modules.

BITRANGE, bit range out of limits

Facility: VDE, OpenVMS Development Environment

BLDALRSTART, build job *job* for stream *stream* already started; build job cannot be started again

Facility: VDE, OpenVMS Development Environment

Explanation: You are attempting to start a build job that cannot be started because it has already been started. A build job can only be started if it is not queued for execution or if it is suspended. As a result, the specified build job could not be started.

User Action: This is an informational message. No user action is required.

BLDALRSTOP, build job *job* for stream *stream* has already stopped; build job cannot be stopped again

Facility: VDE, OpenVMS Development Environment

Explanation: You are attempting to stop a build job that has already been stopped or that has already completed. As a result, the build job could not be stopped.

User Action: This is an informational message. No user action is required.

BLDINIFAIL, build initialization completed with errors

Facility: VDE, OpenVMS Development Environment

Explanation: The build initialization for a build job or a compile or link operation completed with one or more errors. The DCL commands that failed were generated from the build initialization script for the default development stream.

User Action: Correct the errors and resubmit the compilation or build job.

BLDJOBACT, build job already active for stream *stream*; new job not created

Facility: VDE, OpenVMS Development Environment

Explanation: You have tried to create a new build job with the BUILD command when another build job for the same development stream is already active. A build job is ''active'' unless it has completed execution or has been stopped. You cannot have more than one active build job for the

same stream at any one time since that could cause two build jobs to write to the same result directories at the same time.

User Action: Allow the first build job to complete before you create the next one. Alternatively, you can stop the previous build job with the STOP BUILD_JOB command and then create the new build job.

BLDJOBAFTER, build job to be started after *time*

Facility: VDE, OpenVMS Development Environment

Explanation: The build job you are starting will actually start executing after the indicated date and time.

User Action: This is an informational message only. If you do not want the job started after the shown time, stop the job with the STOP BUILD_ JOB command and restart it with a different time using the START BUILD_JOB command.

BLDJOBCRE, build job *job* for stream *stream* has been created

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated build job has been created in the VDE database for the indicated development stream. This message is only issued when you use the /LOG qualifier on the BUILD command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

BLDJOBDEL, build job *job* for stream *stream* deleted from database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated build job has been deleted from the VDE database. This message is only issued when you use the /LOG qualifier on the DELETE BUILD_JOB command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

BLDJOBENT, entry number *entry* submitted to queue *queue*

Facility: VDE, OpenVMS Development Environment

Explanation: A process has successfully been submitted to the indicated batch queue. It received the queue entry number shown in the message.

User Action: This is an informational message. No user action is required.

BLDJOBERR, error queuing build job on queue *queue*; process not queued

Facility: VDE, OpenVMS Development Environment

Explanation: An error occurred when attempting to queue a process on the indicated batch queue on behalf of a build job you are starting. The message that follows gives more information on the reason for the error.

In this case, no more processes are submitted to the indicated queue. Your build job will thus run with fewer processes than your requested.

User Action: This is an informational message only. If you need the displayed batch queue for your build job, stop the build job, correct the error, and resubmit the build job later when the queue is available again.

BLDJOBMOD, build job *job* for stream *stream* modified in database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that information for the indicated build job has been modified in the VDE database. This message is only issued when you use the /LOG qualifier on the MODIFY BUILD_ JOB command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

BLDJOBSIZ, build job *job* for stream *stream* consists of *steps* steps

Facility: VDE, OpenVMS Development Environment

Explanation: This message tells you how many build steps there are in the indicated build job. This message is only issued when you use the /LOG qualifier on the BUILD or START BUILD_JOB command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

BLDJOBSTARTED, build job *job* for stream *stream* started with *processes* processes

Facility: VDE, OpenVMS Development Environment

Explanation: The build job with the shown build-job identification number for the indicated development stream has successfully been started. The message shows the actual number of parallel processes that will execute the build job.

User Action: This is an informational message. No user action is required.

BLDJOBSTARTING, starting build job *job* for stream *stream*

Facility: VDE, OpenVMS Development Environment

Explanation: The build job with the shown build-job identification number for the indicated development stream is about to be started. This message comes out before the processes that will run the build job are submitted to batch queues. Its purpose is to let you know which build job is being started.

User Action: This is an informational message. No user action is required.

BLDJOBSTOP, build job *job* for stream *stream* stopped

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated build job has been stopped. This message is only issued when you use the /LOG qualifier on the STOP BUILD_JOB command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

BLDJOBSUSP, build job *job* for stream *stream* suspended

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated build job has been suspended. A suspended build job can be restarted with the START BUILD_JOB command. This message is only issued when you use the /LOG qualifier on the SUSPEND BUILD_JOB command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

BLDNOTRUN, build job *job* for stream *stream* is not queued or running; build job cannot be suspended

Facility: VDE, OpenVMS Development Environment

Explanation: You are attempting to suspend a build job that is not currently queued for execution or running. As a result, the build job could not be suspended.

User Action: This is an informational message. No user action is required.

BLDSTEPNEED, build step needed to make inputs up-to-date: *step step*

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated build step needs to be performed to make all input modules up-to-date for the compile or link operation you have requested. This situation can occur if, for example, you have modified a source module without compiling it (so that the corresponding object module is no longer up-to-date) and you then request a link that includes the old, non-current object module. This is an informational message to alert you that your compile or link will not produce fully up-to-date outputs, but the requested compile or link will still be performed as you requested.

User Action: If the outputs of the compile or link do not need to be up-to-date, no action is necessary. Otherwise, you can use COMPILE, LINK, or BUILD commands to perform the build steps needed to make all inputs to the original compile or link operation up-to-date with respect to all source changes. You can then repeat that original operation.

BLDSTEPNOTSHO, *steps* additional needed build steps not shown

Facility: VDE, OpenVMS Development Environment

Explanation: If more than 30 build steps are needed to make the inputs to a requested compile or link up-to-date, BLDSTEPNEED messages are only printed for the first 30 such build steps. Then the BLDSTEPNOTSHO message is printed to indicate how many additional build steps are needed, but those build steps are not listed individually.

User Action: If the outputs of the compile or link do not need to be up-to-date, no action is necessary. Otherwise, you can use COMPILE, LINK, or BUILD commands to perform the build steps needed to make all inputs to the original compile or link operation up-to-date with respect to all source changes. You can then repeat that original operation.

CMSBADGEN, bad CMS generation *generation* found for module *module*
possible CMS error

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated bad CMS generation expression was found in an existing CMS library; the cause is probably an error in CMS. The affected generation is not recorded in the VDE database.

User Action: Examine the CMS library and consult a CMS expert.

CMSCLASSEXIST, CMS class *class* already exists for CMS facility *facility*

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated CMS class was found to already exist in the indicated CMS facility, so we do not attempt to create it.

User Action: None.

CMSCRECLASS, CMS error when creating CMS class *class* in facility *facility*

Facility: VDE, OpenVMS Development Environment

Explanation: CMS (Code Management System) has returned an error status when creating a CMS class in the indicated facility. The subsequent print line shows what specific error status CMS returned.

User Action: If the problem appears to be due to an internal VDE error, submit a problem report.

CMSCREELEM, CMS error when creating CMS element for module *module*

Facility: VDE, OpenVMS Development Environment

Explanation: CMS (Code Management System) has returned an error status when creating the CMS element (the delta-file) for the indicated module. The subsequent print line shows what specific error status CMS returned. This error should not occur unless you have insufficient privileges to access the VDE library or the file to be added to that library, or unless you have run out of some resource such as disk space.

User Action: If the error is caused by not having sufficient privileges or resources to perform the action, you must correct the deficiency. Then try

the command again. If the problem appears to be due to an internal VDE error, submit a problem report.

CMSCRELIB, CMS error when creating CMS library for facility *facility*

Facility: VDE, OpenVMS Development Environment

Explanation: CMS (Code Management System) has returned an error status when creating the CMS library for the indicated facility. The subsequent print line shows what specific error status CMS returned. This error should not occur unless you have insufficient privileges to access the VDE library or unless you have run out of some resource such as disk space.

User Action: If the error is caused by not having sufficient privileges or resources to perform the action, you must correct the deficiency. Then try the command again. If the problem appears to be due to an internal VDE error, submit a problem report.

CMSDELELEM, CMS error when deleting element *element*

Facility: VDE, OpenVMS Development Environment

Explanation: CMS (Code Management System) has returned an error status when deleting the indicated module's CMS element. The subsequent print line shows what specific error status CMS returned. This error should not occur unless you have insufficient privileges to access the VDE library or unless you have run out of some resource.

User Action: If the error is caused by not having sufficient privileges or resources to perform the action, you must correct the deficiency. Then try the command again. If the problem appears to be due to an internal VDE error, submit a problem report.

CMSDELGEN, CMS error when deleting generation *generation* (*generation*)

Facility: VDE, OpenVMS Development Environment

Explanation: CMS (Code Management System) has returned an error status when deleting the indicated module generation. The subsequent print line shows what specific error status CMS returned. This error should not occur unless you have insufficient privileges to access the VDE library or unless you have run out of some resource such as disk space.

User Action: If the error is caused by not having sufficient privileges or resources to perform the action, you must correct the deficiency. Then try the command again. If the problem appears to be due to an internal VDE error, submit a problem report.

CMSDUPGEN, duplicate CMS generation *generation* found for module *module*
possible CMS error

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated CMS generation was found more than once for the same CMS element; the cause is probably an error in CMS. The affected generation is not recorded in the VDE database.

User Action: Examine the CMS library and consult a CMS expert.

CMSFETCH, CMS error when fetching CMS element for module *module*

Facility: VDE, OpenVMS Development Environment

Explanation: CMS (Code Management System) has returned an error status when fetching the CMS element for the indicated module. The subsequent print line shows what specific error status CMS returned. This error should not occur unless you have insufficient privileges to access the VDE library or unless you have run out of some resource such as disk space.

User Action: If the error is caused by not having sufficient privileges or resources to perform the action, you must correct the deficiency. Then try the command again. If the problem appears to be due to an internal VDE error, submit a problem report.

CMSINSGEN, CMS error inserting generation *generation* of element *element* into class *class*

Facility: VDE, OpenVMS Development Environment

Explanation: CMS (Code Management System) has returned an error status when inserting a CMS element generation into the indicated class. The subsequent print line shows what specific error status CMS returned.

User Action: If the problem appears to be due to an internal VDE error, submit a problem report.

CMSLIBEMPTY, the CMS library *library* contains no elements

Facility: VDE, OpenVMS Development Environment

Explanation: Since the CMS library specified with the /CMS_LIBRARY qualifier is empty, no modules are created.

User Action: None.

CMSMODCLASS, CMS error modifying the class name *class* to *class* in facility *facility*

Facility: VDE, OpenVMS Development Environment

Explanation: CMS (Code Management System) has returned an error status when modifying a CMS class name in the indicated facility. The subsequent print line shows what specific error status CMS returned.

User Action: If the problem appears to be due to an internal VDE error, submit a problem report.

CMSNOCLASS, CMS class *class* does not exist for facility *facility* Unable to modify CMS class name

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated CMS class does not exist in the indicated facility, so we are not able to modify its name.

User Action: None.

CMSREMGEN, CMS error removing element *element* from class *class*

Facility: VDE, OpenVMS Development Environment

Explanation: CMS (Code Management System) has returned an error status when removing a CMS element generation from the indicated class. The subsequent print line shows what specific error status CMS returned.

User Action: If the problem appears to be due to an internal VDE error, submit a problem report.

CMSREPLACE, CMS error when replacing CMS element for module *module*

Facility: VDE, OpenVMS Development Environment

Explanation: CMS (Code Management System) has returned an error status when replacing the CMS element for the indicated module. The subsequent print line shows what specific error status CMS returned. This error should not occur unless you have insufficient privileges to access the VDE library or unless you have run out of some resource such as disk space.

User Action: If the error is caused by not having sufficient privileges or resources to perform the action, you must correct the deficiency. Then try the command again. If the problem appears to be due to an internal VDE error, submit a problem report.

CMSRESERVE, CMS error when reserving CMS element for module *module*

Facility: VDE, OpenVMS Development Environment

Explanation: CMS (Code Management System) has returned an error status when reserving the CMS element for the indicated module. The subsequent print line shows what specific error status CMS returned. This error should not occur unless you have insufficient privileges to access the VDE library or unless you have run out of some resource such as disk space.

User Action: If the error is caused by not having sufficient privileges or resources to perform the action, you must correct the deficiency. Then try the command again. If the problem appears to be due to an internal VDE error, submit a problem report.

CMSSETLIB, CMS error when setting CMS library for facility *facility*

Facility: VDE, OpenVMS Development Environment

Explanation: CMS (Code Management System) has returned an error status when making the CMS library for the indicated facility the current CMS library. The subsequent print line shows what specific error status CMS returned. This error should not occur unless you have insufficient privileges to access the VDE library.

User Action: If the error is caused by not having sufficient privileges to perform the action, you must correct the deficiency. Then try the command again. If the problem appears to be due to an internal VDE error, submit a problem report.

CMSSHOWCLASS, CMS error when retrieving CMS class information in library *library*

Facility: VDE, OpenVMS Development Environment

Explanation: CMS (Code Management System) has returned an error status when retrieving CMS class information for the indicated CMS library. The subsequent print line shows what specific error status CMS returned.

User Action: If the problem appears to be due to an internal VDE error, submit a problem report.

CMSSHOWDESGEN, CMS error when retrieving CMS generation of element *element*

Facility: VDE, OpenVMS Development Environment

Explanation: CMS (Code Management System) has returned an error status when retrieving CMS generation information for the indicated CMS element. The subsequent print line shows what specific error status CMS returned.

User Action: If the problem appears to be due to an internal VDE error, submit a problem report.

CMSSHOWELEM, CMS error when retrieving CMS element in library *library*

Facility: VDE, OpenVMS Development Environment

Explanation: CMS (Code Management System) has returned an error status when retrieving CMS element information for the indicated CMS library. The subsequent print line shows what specific error status CMS returned.

User Action: If the problem appears to be due to an internal VDE error, submit a problem report.

CMSSHOWHIST, CMS error when retrieving CMS history in library *library*

Facility: VDE, OpenVMS Development Environment

Explanation: CMS (Code Management System) has returned an error status when retrieving CMS history information for the indicated CMS library. The subsequent print line shows what specific error status CMS returned.

User Action: If the problem appears to be due to an internal VDE error, submit a problem report.

CMSSHOWRES, CMS error when retrieving CMS reservations in library *library*

Facility: VDE, OpenVMS Development Environment

Explanation: CMS (Code Management System) has returned an error status when retrieving CMS reservation information for the indicated

CMS library. The subsequent print line shows what specific error status CMS returned.

User Action: If the problem appears to be due to an internal VDE error, submit a problem report.

CMSSTART, starting retrieval of CMS information for facility *facility*

Facility: VDE, OpenVMS Development Environment

Explanation: VDE is starting to retrieve module, generation, and reservation information from the exiting CMS library for the indicated facility. After this information is retrieved, it will be included in the VDE database.

User Action: None.

CMSVARLET, all CMS variant letters in use; none left for module *module*

Facility: VDE, OpenVMS Development Environment

Explanation: All CMS variant letters up to "Z" are already in use for the indicated module. Hence the current REPLACE operation cannot be performed.

User Action: Submit a problem report. This is a very unlikely situation, but there is no known workaround. Additional VDE code is needed to handle this situation.

CMSVERGEN, CMS error reading class membership list when verifying generation *generation* (*spec*)

Facility: VDE, OpenVMS Development Environment

Explanation: CMS (Code Management System) has returned an error status when reading the class membership list for the indicated module generation. The subsequent print line shows what specific error status CMS returned. This error should not occur unless you have insufficient privileges to access the VDE library or unless you have run out of some resource such as disk space.

User Action: If the error is caused by not having sufficient privileges or resources to perform the action, you must correct the deficiency. Then try the command again. If the problem appears to be due to an internal VDE error, submit a problem report.

COMMAND, command *command*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you of the particular command that was (or was not) executed. This message is normally associated with other messages or errors of some sort.

User Action: If unable to resolve the basic error, please contact VDE support for assistance. Please include any information that may be useful or relevent in duplicating the problem.

COMMIT, database transaction has successfully committed

Facility: VDE, OpenVMS Development Environment

Explanation: The database transaction associated with the current command has successfully committed, thus ensuring that all changes have been written to the VDE database.

User Action: Feel relieved.

COMNOTDEF, command "*command-text*" not defined

Facility: VDE, OpenVMS Development Environment

Explanation: DELETE/COMMAND issues this message when the command you request to delete was not defined.

User Action: Use SHOW SYMBOL/DEFINED to display all defined commands.

COMPFAIL, compilation of *module* completed with errors

Facility: VDE, OpenVMS Development Environment

Explanation: The compile step for the indicated module completed with one or more errors. The compiler's output (on your terminal or in the step's log file) gives more detail on the errors it detected.

User Action: Correct the errors and resubmit the compilation or build job.

COMPSUCC, compilation of *module* completed successfully

Facility: VDE, OpenVMS Development Environment

Explanation: The compilation of the indicated module completed successfully.

User Action: None.

COMSTEPADDED, compile step for *module* added to database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated compile step has been added to the VDE database. This message is only issued when you use the /LOG qualifier on the CREATE STEP command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

COMSTEPDEL, compile step for module *module* deleted from database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the compile step for the indicated module has been deleted from the VDE database. This means that all build dependency information for that build step has been deleted.

This message is only issued when you use the /LOG qualifier on the DELETE STEP command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

COMSTEPMOD, compile step for module *module* modified in database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that build dependency information for the indicated compile step has been modified in the VDE database. This message is only issued when you use the /LOG qualifier on the MODIFY STEP command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

COMTERMEOF, command terminated by CTRL/Z or end-of-file on input

Facility: VDE, OpenVMS Development Environment

Explanation: The current command has been terminated because the user entered a CTRL/Z in response to a query for additional information needed to execute the command. Alternatively, an end-of-file may have been encountered in the current input file in response to such a query. This error can also appear if an attempt is made to use DCL command procedure with a VDE kept subprocess.

User Action: Not an error since a user may intentionally terminate commands in this way. To execute the original command, reenter the command and supply all required information. If using a DCL command procedure to issue VDE commands, disable the use of the VDE kept subprocess, or switch to a VDE command procedure.

CONCURRES, concurrent reservation *reservation* of *module* by user *user* already present

Facility: VDE, OpenVMS Development Environment

Explanation: You are reserving a module which is already reserved by another user (possibly yourself). That reservation allows concurrent reservations.

User Action: This is an informational message which appears when you are creating a concurrent reservation. This message informs you of each already existing reservation for the same module that covers at least one common stream.

CONFACC, unable to access notes conference

Facility: VDE, OpenVMS Development Environment

Explanation: Check the notes conference for general accessability, and—if conference membership is needed—determine if you are a member of the conference from the node on which you are running VDE.

User Action: See the accompanying message(s).

CONFLICT, conflicting qualifers and/or keywords specified

Facility: VDE, OpenVMS Development Environment

Explanation: This message is issued when VDE detects conflicting qualifiers and/or keywords within the same command.

User Action: Verify the intention of your command and see if the use of a specific qualifier or keyword may exclude the use of one of the other qualifiers and/or keywords. See VDE Reference Manual for assistance with the command syntax.

CONFMEM, check conference file accessability and membership

Facility: VDE, OpenVMS Development Environment

Explanation: Check the notes conference for general accessability, and—if conference membership is needed—determine if you are a member of the conference from the node on which you are running VDE.

User Action: See the accompanying message(s).

CONFNAM, conference file name *name*

Facility: VDE, OpenVMS Development Environment

Explanation: This is the name of the notes conference.

User Action: See the accompanying message(s).

CONKEYATR, conflicting key attributes *key-attribute* and *key-attribute*

Facility: VDE, OpenVMS Development Environment

Explanation: The recently issued DEFINE KEY command has conflicting qualifiers such as /NOSTATE and /LOCK_STATE.

User Action: Determine correct applicability and reissue the command.

CONQUAL, conflicting qualifiers */qualifier* and */qualifier*

Facility: VDE, OpenVMS Development Environment

Explanation: The above mentioned qualifiers are mutually exclusive.

User Action: Verify the intention of your command and see if the use of a specific qualifier may mutually exclude the use of other qualifiers.

CONRESDEL, concurrent reservation *reservation* of *module* for user *user* has been deleted

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated concurrent reservation has been deleted from the VDE database because you replaced the same module.

User Action: None. The affected users will be informed that their reservations are deleted. They should reserve the module again and merge in their changes.

CONTINUE, attempting to continue

Facility: VDE, OpenVMS Development Environment

Explanation: One or more quotas are below the recommended values for VDE. VDE will attempt to continue.

User Action: Contact system management and request that the quotas associated with the current process be compared to the values recommended for use with VDE.

COPSTEPADDED, copy step for *step* added to database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated copy step has been added to the VDE database. This message is only issued when you use the /LOG qualifier on the CREATE STEP command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

COPSTEPDEL, copy step for module *module* deleted from database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the copy step for the indicated module has been deleted from the VDE database. This means that all build dependency information for that build step has been deleted. This message is only issued when you use the /LOG qualifier on the DELETE STEP command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

COPSTEPMOD, copy step for module *module* modified in database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that build dependency information for the indicated copy step has been modified in the VDE database. This message is only issued when you use the /LOG qualifier on the MODIFY STEP command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

COPYFAIL, copy of *module* completed with errors

Facility: VDE, OpenVMS Development Environment

Explanation: The copy step for the indicated module completed with one or more errors. The step's output (on your terminal or in the step's log file) gives more detail on the errors it detected.

User Action: Correct the errors and resubmit the copy step or build job.

COPYSUCC, copy of *module* completed successfully

Facility: VDE, OpenVMS Development Environment

Explanation: The copy step for the indicated module completed successfully.

User Action: None.

CREDIRFAI, create directory failed

Facility: VDE, OpenVMS Development Environment

Explanation: A directory specification on a VDE command is not syntactically correct. This message is also printed if a logical name (such as VDE$LIBRARY) that is supposed to be bound to a directory specification is not bound to a syntactically correct directory specification.

User Action: Reenter the VDE command with a valid directory specification or define the logical name to be bound to a valid directory specification.

CRELNMFAI, failed to define logical name *lnmdsc*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you of the particular logical name creation that failed. This message is normally associated with other messages or errors of some sort.

User Action: If unable to resolve the basic error, please contact VDE support for assistance. Please include any information that may be useful or relevent in duplicating the problem.

CTXADDED, context *context* added to the database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated context definition has been added to the VDE database. This message is only issued when you use the /LOG qualifier on the CREATE CONTEXT command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

CTXDEL, context *context* deleted from the database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the definition for the indicated context has been deleted from the VDE database. This message is only issued when you use the /LOG qualifier on the DELETE CONTEXT command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

CTXDELARCH, context ## contains deleted architecture *architecture*

Facility: VDE, OpenVMS Development Environment

Explanation: You have attempted to set the indicated context, but the default architecture specified by the context has been deleted. As a result, the context cannot be set.

User Action: Redefine the context to reference an architecture that has not been deleted. Then reissue the command.

CTXDELFAC, context *context* contains deleted facility *facility*

Facility: VDE, OpenVMS Development Environment

Explanation: You have attempted to set the indicated context, but the default facility specified by the context has been deleted. As a result, the context cannot be set.

User Action: Redefine the context to reference a facility that has not been deleted. Then reissue the command.

CTXDELSTRM, context *context* contains deleted stream *context*

Facility: VDE, OpenVMS Development Environment

Explanation: You have attempted to set the indicated context, but the default stream specified by the context has been deleted. As a result, the context cannot be set.

User Action: Redefine the context to reference a stream that has not been deleted. Then reissue the command.

CTXEXISTS, context *context* already exists in database; context not changed

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated context name was specified on a CREATE CONTEXT command, but an entry for that context is already present in the database. Hence a new entry was not created for that context. Alternatively, the indicated name was specified as the new name for an existing context on the MODIFY CONTEXT command, but another context already has the requested new name. Hence the name of the existing context could not be changed to the new name.

User Action: If you entered the wrong context name, reenter the command with the correct name. If you entered the right context name, use the SHOW CONTEXT command to display the database information for that context. If you are not satisfied with that information, use the MODIFY CONTEXT command to change it.

CTXMOD, information for context *context* modified in the database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that information for the indicated context has been modified in the VDE database. This message is

only issued when you use the /LOG qualifier on the MODIFY CONTEXT command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

CTXNOTFOUND, context *context* not found in database; parameter ignored

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated context name was entered on a SHOW CONTEXT command, but no such context was found in the VDE database.

User Action: This is not necessarily an error; the message simply tells you that the indicated context does not exist in the database.

CTXSYNTAX, invalid context name syntax: *name*

Facility: VDE, OpenVMS Development Environment

Explanation: The name or format specified for the context name is not using valid syntax.

User Action: Respecify the context name using only alphanumeric, underscore, dollar sign and hyphen characters.

CULPRIT, reserved by user *user*

Facility: VDE, OpenVMS Development Environment

Explanation: the module is reserved by the specified user. The reservation was requested in the current stream, or in an ancestor of the current stream.

User Action: This message is informational, and should accompany one or more other error messages.

DATETOOLONG, date-time value is too long; maximum length is 24: *date...*

Facility: VDE, OpenVMS Development Environment

Explanation: A date-time value entered as a command parameter is too long because it exceeds the maximum length indicated in the message.

User Action: Reissue the command with a shorter date-time value.

DBCNV, converting database format from *oldver* to *newver*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to display the status of a database conversion operation.

User Action: No action is required.

DBCREATED, database *db* has been created

Facility: VDE, OpenVMS Development Environment

Explanation: This message indicates that a new library database, and associated database files, directories, and CMS libraries has been created.

User Action: None required.

DBFMT, database format is *ver*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to display the current version of a database, or the status of a database conversion operation.

User Action: No action is required.

DBGCMSGEN, at *codept*, letflg *bool*, item1 *itm1*, item2 *did*

Facility: VDE, OpenVMS Development Environment

Explanation: This message displays various internal process-related values used within VDE. This message should appear only during VDE testing, and should be displayed only by debugging versions of VDE.

User Action: Contact VDE support for assistance, as the version of VDE in use contains debug-only images.

DBGCMSTXT, at *codept*, expected *gen*

Facility: VDE, OpenVMS Development Environment

Explanation: This message displays various internal process-related values used within VDE. This message should appear only during VDE testing, and should be displayed only by debugging versions of VDE.

User Action: Contact VDE support for assistance, as the version of VDE in use contains debug-only images.

DBGDB, debugging database: *param1*, *param2*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used during development and displays where the behaviour of a debugging or prototype VDE image database operation differs from that of a production VDE image.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

DBGLNMBAT, debug logical name VDE$BATCH_MODE: *mode*

Facility: VDE, OpenVMS Development Environment

Explanation: This message indicates the batch mode setting of VDE, and should appear only during VDE testing. This message is displayed only by debug versions of VDE, and this logical name is translated only by debug versions.

User Action: Contact VDE support for assistance, as the version of VDE in use contains debug-only images.

DBGLNMDAT, debug logical name VDE$DATE_TIME: *mode*

Facility: VDE, OpenVMS Development Environment

Explanation: This message indicates the date-time setting of VDE, and should appear only during VDE testing. This message is displayed only by debug versions of VDE, and this logical name is translated only by debug versions.

User Action: Contact VDE support for assistance, as the version of VDE in use contains debug-only images.

DBGLNMUSR, debug logical name VDE$USERNAME: *mode*

Facility: VDE, OpenVMS Development Environment

Explanation: This message indicates the username setting of VDE, and should appear only during VDE testing. This message is displayed only by debug versions of VDE, and this logical name is translated only by debug versions.

User Action: Contact VDE support for assistance, as the version of VDE in use contains debug-only images.

DBGNOLNM, unable to translate *logicalname*

Facility: VDE, OpenVMS Development Environment

Explanation: VDE was unable to translate a mandatory logical name.

User Action: Contact VDE support for assistance, as the version of VDE running contains debug-only images. Correct the error in the date-time logical name value and reissue the command.

DBGQUALMSK, qualifier mask *qualmsk1 qualmsk2*

Facility: VDE, OpenVMS Development Environment

Explanation: This message displays the contents of an internal bitmask during program execution.

User Action: Contact VDE support for assistance, as the version of VDE in use contains debugging images.

DBGWAYPT, debug waypoint *waypoint* reached

Facility: VDE, OpenVMS Development Environment

Explanation: This message displays a waypoint, and the associated waypoint value, encountered during program execution.

User Action: Contact VDE support for assistance, as the version of VDE in use contains debugging images.

DBGXLAC, debug *waypoint* reached; text: *text*

Facility: VDE, OpenVMS Development Environment

Explanation: This message displays a waypoint, and the associated waypoint value, encountered during program execution.

User Action: Contact VDE support for assistance, as the version of VDE in use contains debugging images.

DBGXLXL, debug *waypoint* reached; value: *value*

Facility: VDE, OpenVMS Development Environment

Explanation: This message displays a waypoint, and the associated waypoint value, encountered during program execution.

User Action: Contact VDE support for assistance, as the version of VDE in use contains debugging images.

DCLBUFOVFL, DCL command line being generated is too long for buffer

Facility: VDE, OpenVMS Development Environment

Explanation: Some command which generates a DCL command file to perform part of its actions generated a DCL command line that is too long for its buffer; this normally means that the line is excessively long since the buffer size is ample for normal use.

User Action: Change the command line specification in the script for it.

DCLERROR, error status returned by spawned DCL *DCL* command

Facility: VDE, OpenVMS Development Environment

Explanation: VDE has run a DCL command in a spawned subprocess and that command has returned an error status. The command's status is displayed by the following status line.

User Action: Determine the cause of the problem, correct it, and retry the VDE command.

DCLTRUNC, DCL command line from script truncated; line too long

Facility: VDE, OpenVMS Development Environment

Explanation: A script has generated a DCL command line that is too long and has been truncated. Since the editing buffer is of ample size for normal use, the script tried to generate an excessively long DCL command line.

User Action: Correct the script to generate a shorter DCL command line.

DEFARCH, architecture *architecture* not deleted; you cannot delete default architecture

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated architecture, namely the current default architecture, has not been deleted from the database. You are not allowed to delete the default architecture. This prohibition prevents you from accidentally deleting all architectures,

especially when using a wildcarded architecture name specification. There is thus no way to delete all architectures from the database. (If you could delete all architectures, the library would become useless since it could not contain any modules.)

User Action: If you want to delete a given architecture from the database, set your default architecture to another architecture that you want to keep. Then issue the DELETE ARCHITECTURE command.

DEFKEY, *state-name* key *key-name* has been defined

Facility: VDE, OpenVMS Development Environment

Explanation: Issued by DEFINE/KEY to inform the user that the specific key definition has been bound.

User Action: If you did not wish to see this message, issue /NOLOG with your DEFINE command.

DEFKEYERR, error defining *state-name* key *key-name*

Facility: VDE, OpenVMS Development Environment

Explanation: VDE uses the SMG screen management facilities for the purpose of managing key bindings. In this instance, the call to SMG$ADD_KEY_DEF returned a bad status.

User Action: First, try to interpret the reason for the SMG error. If it can be understood, then try to see if you may fix it locally. Otherwise, please contact VDE support for assistance.

DELKEY, *state-name* key *key-name* has been deleted

Facility: VDE, OpenVMS Development Environment

Explanation: Issued by DELETE/KEY to inform the user that the specific key definition has been deleted.

User Action: If you do not wish to see this message, issue /NOLOG with your DELETE command.

DELKEYERR, error in processing DELETE/KEY command

Facility: VDE, OpenVMS Development Environment

Explanation: VDE uses the screen management facilities, SMG, for the purpose of managing key bindings. In this instance, a call to either SMG$LIST_KEY_DEFS or SMG$DELETE_KEY_DEF has returned a bad status.

User Action: First, try to interpret the reason for the SMG error. If it can be understood, then try to see if you may fix it locally. Otherwise, please contact VDE support for assistance.

DEPNOTUPDATE, dependency information not updated for build step

Facility: VDE, OpenVMS Development Environment

Explanation: The dependency information for the most recent build step has not been updated in the VDE database because the build step did not automatically collect dependency information. Hence the existing dependency information is kept without change in the database.

User Action: None.

DEPUPDATE, dependency information updated for build step

Facility: VDE, OpenVMS Development Environment

Explanation: The dependency information for the most recent build step has been updated in the VDE database based on dependency data collected during the execution of the build step.

User Action: None.

DEVOSYNCH, executing synch command: *dclcmd*

Facility: VDE, OpenVMS Development Environment

Explanation: This message displays the synchronization command being spawned by VDE. This message should appear only during VDE testing, and should be displayed only by debugging versions of VDE.

User Action: Contact VDE support for assistance, as the version of VDE in use contains debug-only images.

DIRADDED, directory specification *spec* added to the database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated directory specification has been added to the VDE database. This message is only issued when you use the /LOG qualifier on the SET DIRECTORY command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

DIRCREATED, directory *directory* has been created

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated directory has just been created as your VDE scratch directory.

User Action: Check the newly created directory specification and delete the directory if you do not wish to keep it. To change the default VDE scratch directory specification, define VDE$SCRATCH.

DIRDEL, directory specification *spec* deleted from database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated stream root or facility root directory specification has been deleted from the VDE database. This message is only issued when you use the /LOG qualifier on the CANCEL DIRECTORY command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

DIREXPBAD, directory filename expansion bad

Facility: VDE, OpenVMS Development Environment

Explanation: internal state information for VDE$_INVDIRSPEC error.

User Action: Correct the cause of the VDE$_INVDIRSPEC error.

DIREXPFAI, directory filename expansion failed

Facility: VDE, OpenVMS Development Environment

Explanation: internal state information for VDE$_INVDIRSPEC error.

User Action: Correct the cause of the VDE$_INVDIRSPEC error.

DIRFILFAI, directory filename conversion failed

Facility: VDE, OpenVMS Development Environment

Explanation: internal state information for VDE$_INVDIRSPEC error.

User Action: Correct the cause of the VDE$_INVDIRSPEC error.

DIRFILNOTDEL, unable to delete directory file *directory*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated directory file was not deleted for some reason, such as insufficient privileges.

User Action: Use DCL commands to check the security on the directory file, and take the appropriate action to delete it if possible.

DIRNOTFOUDCL, directory location not found for script's DCL command file

Facility: VDE, OpenVMS Development Environment

Explanation: The subdirectory in the VDE library that should be used for the DCL command file that the current script generates is not found in the database. This message indicates that some required subdirectory is not present in the directory tree for the current development stream.

User Action: Create the directory tree or correct the directory structure for the current stream or stream and facility. Then reissue the command that invokes the script.

DIRNOTFOULOG, directory location not found for build-step log file

Facility: VDE, OpenVMS Development Environment

Explanation: The subdirectory in the VDE library that should be used for the log file for the current build step is not found in the database. This message indicates that some required subdirectory is not present in the directory tree for the current development stream.

User Action: Create the directory tree or correct the directory structure for the current stream or stream and facility. Then reissue the command that executes the build step.

DIRNOTFOUND, directory specification to cancel not found in database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the directory specification you are trying to cancel with the CANCEL DIRECTORY command does not exist in the VDE database. This message is only issued when you use the /LOG qualifier on the CANCEL DIRECTORY command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

DIRNOTFOUSCRP, directory location not found by script function *function*

Facility: VDE, OpenVMS Development Environment

Explanation: The current script has invoked the indicated script function (such as $DIR or $FACROOT), but the directory location specified with that function invocation is not found in the VDE database. This may mean that the directory tree for the specified stream or facility has not been created, or that the function invocation specifies a directory symbol that is not used for that stream and facility.

User Action: Create the directory tree for the facility or stream, or correct the directory symbol name in the function call. Then reissue the command that invokes the script.

DIRSPECREQ, directory specification required for stream *stream*

Facility: VDE, OpenVMS Development Environment

Explanation: A root directory specification for the indicated stream is required on the CREATE STREAM command but was not specified. You need to use the /DIRECTORY qualifier to specify the root directory for the stream, or you need to enter that directory specification when prompted for it. This message is only given if an earlier SET DIRECTORY/PROMPT command has specified that the CREATE STREAM command should prompt for the directory specification if the /DIRECTORY qualifier is missing.

User Action: Enter a directory specification for the stream root directory when prompted for it or reissue the CREATE STREAM command with the /DIRECTORY qualifier.

DIRTOOLONG, directory specification is too long; maximum length is 255

Facility: VDE, OpenVMS Development Environment

Explanation: A directory specification entered as a command parameter is too long because it exceeds the maximum length indicated in the message.

User Action: Reissue the command with a shorter directory specification.

DIRTREFAI, directory tree conversion failed

Facility: VDE, OpenVMS Development Environment

Explanation: internal state information for VDE$_INVDIRSPEC error.

User Action: Correct the cause of the VDE$_INVDIRSPEC error.

DIRTRENOTDEL, not all files deleted in directory tree *tree*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that not all files in the indicated directory tree were deleted for some reason, such as insufficient privileges.

User Action: Use DCL commands to check the security on the remaining files in the directory tree, and take the appropriate action to delete them if possible.

DUPLNAME, duplicate name in list not allowed: *name*

Facility: VDE, OpenVMS Development Environment

Explanation: You have entered the same name twice in a comma list where such duplicates are not allowed. Possibly, you may have mistyped one of the names.

User Action: Correct the error and reissue the command without any duplicates.

EDITUNDEF, your editor is undefined; editor cannot be invoked

Facility: VDE, OpenVMS Development Environment

Explanation: The command you have entered is attempting to invoke an editor to read or edit a file, but you have not specified what editor to invoke. You must specify the editor you want VDE to use by entering the SET EDITOR command.

User Action: Use the SET EDITOR command to define the editor you want VDE to invoke for all commands to put you in an editor.

EDITUPDATE, default editor definition updated in the VDE database

Facility: VDE, OpenVMS Development Environment

Explanation: Your default editor has been updated in the VDE database. The default editor is the editor that is initially defined each time you start a VDE session.

User Action: None. This is an informational message only.

ELENOTFOUNDCMS, element *element* not found in CMS library

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated module generation does not have an element in the CMS library.

User Action: Use the VERIFY GENERATION/RECOVER command to attempt recovery of the missing generation or locate missing element from a backup and manually insert the element and generations in the database.

ERRASSIGN, the attempt to acquire an I/O channel for VDE failed

Facility: VDE, OpenVMS Development Environment

Explanation: A bad status was returned from a SYS$ASSIGN service call. VDE needs to acquire I/O channels to do I/O. In this case, VDE failed to acquire such a channel.

User Action: Check your process's quotas. If this problem cannot be resolved locally, please contact VDE support for assistance.

ERRCRELNM, unable to create a logical name

Facility: VDE, OpenVMS Development Environment

Explanation: VDE creates logical names for input and output redirection.

User Action: Check your process's quotas. If this problem cannot be resolved locally, please contact VDE support for assistance.

ERRDEASSIGN, attempt to deassign an I/O channel acquired by VDE failed

Facility: VDE, OpenVMS Development Environment

Explanation: VDE wanted to deassign an I/O channel that was acquired. This error notes the failure of the SYS$DASSGN system service, probably due to an invalid channel.

User Action: Please contact VDE support for assistance.

ERRDIRSPEC, error in directory or file specification *file-name*

Facility: VDE, OpenVMS Development Environment

Explanation: There is a syntax error or other error in the indicated directory or file specification. The RMS message that follows explains the error in more detail.

User Action: Correct the error in the directory or file specification and reissue the command.

ERRFAO, unable to format output string

Facility: VDE, OpenVMS Development Environment

Explanation: An error was returned from a call to SYS$FAO.

User Action: Please contact VDE support for assistance.

ERRQIOW, error from SYS$QIOW

Facility: VDE, OpenVMS Development Environment

Explanation: A bad status was returned from a call to SYS$QIOW.

User Action: Please contact VDE support for assistance.

ERRSMG, error returned from a call to the Screen Management Facility (SMG)

Facility: VDE, OpenVMS Development Environment

Explanation: A bad status was returned from a call to a SMG routine. This could be a result of any number of things which may or may not be a VDE problem.

User Action: Check the setup of the terminal which might cause SMG some problem. If the error still cannot be explained or resolved, please contact VDE support for assistance.

ERRWRIFIL, error writing file *file-name*

Facility: VDE, OpenVMS Development Environment

Explanation: The output could not be written to the appropriate file.

User Action: Determine if there is any external interference with RMS writing to the specified file. If no interference can be found, please contact VDE support for assistance.

EXARANGE, hexadecimal number not valid; valid range is *number* to *number*

Facility: VDE, OpenVMS Development Environment

Explanation: Illegal hexadecimal range provided as input.

User Action: Verify that the provided hexadecimal range is in an accepted format.

EXPGENDIFF, expected generation expression *expression* differs from actual *expression*

Facility: VDE, OpenVMS Development Environment

Explanation: When you reserved the module you are replacing, the RESERVE command displayed the expected replace-time CMS generation expression of the new module generation. However, the actual CMS generation expression that VDE assigned during the current replacement differs from that expected generation expression. This situation is normal because VDE cannot predict the actual replace-time CMS generation expression when you reserve the module; it can only tell you the most likely generation expression. By the time you replace the module, various circumstances can change the appropriate generation expression to use.

User Action: None. This message is informational only. However, if you used the expected CMS generation expression in code comments, you may have to change those comments.

EXPGENEXPR, expected generation expression at replacement is *generation*

Facility: VDE, OpenVMS Development Environment

Explanation: When you replace the module you just reserved, the new generation of the module is expected to have the indicated CMS generation expression. You may use this generation expression as an ''audit trail identifier'' in the comments in your code if your project uses that convention. The indicated generation expression may differ from the one that VDE actually assigns when you replace the module, but only in certain unusual circumstances.

User Action: None. This message is informational only.

EXPGENLONG, expected CMS generation expression is too long

Facility: VDE, OpenVMS Development Environment

Explanation: The expected replace-time CMS generation expression for the module you are reserving will be longer than 16 characters, which is longer than CMS allows. As a result, you cannot reserve the module.

User Action: This error indicates that you have reserved and replaced modules on too many nested non-main streams. You should create new streams off the main development stream and place the offending generations there so that they have CMS generation expressions of modest size. Even though you cannot reserve the generation in question, you can still fetch it out and replace it into the new streams.

FACADDED, facility *facility* added to the database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated facility definition has been added to the VDE database. This message is only issued when you use the /LOG qualifier on the CREATE FACILITY command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

FACDEL, facility *facility* deleted from the database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the definition for the indicated facility has been deleted from the VDE database. This message is only issued when you use the /LOG qualifier on the DELETE FACILITY command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

FACEXISTS, facility *facility* already exists in database

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated facility name was specified on a CREATE FACILITY command, but an entry for that facility is already present in the database. Alternatively, the indicated name was specified as the new name for an existing facility using the MODIFY FACILITY command, but another facility already has the requested new name. In either case, the command fails and the facility is not created or modified.

User Action: If you entered the wrong facility name, reenter the command with the correct name. If you entered the right facility name, use the SHOW FACILITY command to display the database information for that facility. If you are not satisfied with that information, use the MODIFY FACILITY command to change it.

FACMOD, information for facility *facility* modified in the database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that information for the indicated facility has been modified in the VDE database. This message is only issued when you use the /LOG qualifier on the MODIFY FACILITY command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

FACNOTDEL, facility *facility* not deleted

Facility: VDE, OpenVMS Development Environment

Explanation: You are attempting to delete a facility that cannot be deleted. Additional messages are normally included with this message, and these messages indicate why the facility cannot be deleted.

User Action: Resolve the problem(s) flagged by the other error messages, and reenter the DELETE FACILITY command.

FACNOTFOUND, facility *facility* not found in database; parameter ignored

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated facility name was entered on a SHOW FACILITY command, but no such facility was found in the VDE database.

User Action: This is not necessarily an error; the message simply tells you that the indicated facility does not exist in the database.

FACNOTMT, facility contains modules

Facility: VDE, OpenVMS Development Environment

Explanation: The specified facility contains modules.

User Action: The requested operation is prohibited due to the presence of modules in the facility.

FACSYNTAX, invalid facility name syntax: *name*

Facility: VDE, OpenVMS Development Environment

Explanation: The name or format specified for the facility name is not using valid syntax.

User Action: Respecify the facility name using only alphanumeric, underscore, dollar sign and hyphen characters.

FAMILY, unable to deallocate: siblings or children present

Facility: VDE, OpenVMS Development Environment

Explanation: Something deep within VDE has attempted to deallocate a data structure that has siblings or children.

User Action: Please contact VDE support for assistance. Please include any information that may be useful or relevent in duplicating the problem.

FCBERR, internal VDE FCB error in *VDE-internal-identifier*

Facility: VDE, OpenVMS Development Environment

Explanation: VDE has detected an internal inconsistency.

User Action: Please contact VDE support for assistance. Please include any information that may be useful or relevent in duplicating the problem.

FCBWRONG, internal error; *access* access failed in *VDE-internal-identifier*

Facility: VDE, OpenVMS Development Environment

Explanation: VDE has detected an internal inconsistency.

User Action: Please contact VDE support for assistance. Please include any information that may be useful or relevent in duplicating the problem.

FETCHED, generation *generation* fetched from library

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated module and generation has been fetched from the current library. This message is only issued when you use the /LOG qualifier on the FETCH command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

FILCOPLIB, file *file* copied to library

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated file has been copied into the VDE library by the REPLACE command. When this is done, the file has been included in the appropriate CMS element, but has not yet been recorded in the VDE database. This message is only issued when you use the /LOG qualifier on the REPLACE command for an immediate replacement.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

FILCOPSTAG, file *file* copied to staging area

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated file has been copied into a VDE staging area by the REPLACE command. This message is only issued when you use the /LOG qualifier on the REPLACE command for a queued replacement.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

FILENOTFOUND, file not found: *file*

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated file was requested through a VDE command, but could not be found. Hence the current command cannot perform the requested action.

User Action: Do without the file or determine why the file is missing and correct the problem.

FILQUE, print file queued to *queue-name*

Facility: VDE, OpenVMS Development Environment

Explanation: This this message informs the user where the file was queued for printing.

User Action: Ensure that the file is placed on the appropriate queue.

FILREADERR, unable to read file *file*

Facility: VDE, OpenVMS Development Environment

Explanation: An error occurred when trying to read a line of text from an input file. A more specific error message follows this message.

User Action: Examine the error message that follows to diagnose the reason for the error. Then correct the problem accordingly.

FILWRITERR, unable to write file *file*

Facility: VDE, OpenVMS Development Environment

Explanation: An error occurred when trying to write a line of text to an output file. A more specific error message follows this message.

User Action: Examine the error message that follows to diagnose the reason for the error. Then correct the problem accordingly.

FIRGENNODEL, generation *generation* is first generation of module; cannot be deleted

Facility: VDE, OpenVMS Development Environment

Explanation: You are trying to delete the first generation of a module with the DELETE GENERATION command, which VDE does not allow. If

you want to delete all generations of a module, you must delete the entire module with the DELETE MODULE command.

User Action: If you mistyped the generation number, reenter it correctly. If you want to delete all generations of the module, use the DELETE MODULE command.

FIRNUMMAY, first number of a range may not be less than *number*

Facility: VDE, OpenVMS Development Environment

Explanation: Some integers provided in the command line must be greater than a given "n".

User Action: Verify that the appropriate "n" was given in the command line.

FIRNUMRAN, first number of a range must be less than the second

Facility: VDE, OpenVMS Development Environment

Explanation: When providing a range in the format n[:m], "n" must be less than "m".

User Action: Correct the range description and reissue the command.

FLAVORPIDS, flavor *flavor*, pid *pid*, master *mpid*, parent *ppid*

Facility: VDE, OpenVMS Development Environment

Explanation: This message displays various internal process-related values used within VDE. This message should appear only during VDE testing, and should be displayed only by debugging versions of VDE.

User Action: Contact VDE support for assistance, as the version of VDE in use contains debug-only images.

FOLDADDED, fold record *record* for module *module* added to database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated module has had a fold record created in the current VDE database. This message is only issued when you use the /LOG qualifier on the CREATE FOLD command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

FOLDDEL, fold record *record* for module *module* has been deleted

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated module has had a fold record deleted from the current VDE database. This message is only issued when you use the /LOG qualifier on the DELETE FOLD command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

FOLDEXP, fold record for *fold* expected for stream *stream*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that VDE expects to generate a fold record for the indicated stream. The fold record will be generated when this module change is approved and performed by the project leader—when the queued replacement is performed. Because there may be a delay before the replacement is performed by the project leader and because the project leader may reject the replacement, the fold record will not immediately appear via commands such as SHOW REPLACEMENT. The fold will be generated when VDE sends you the notification that the replacement has been performed. This message is only issued when you use the /LOG qualifier on the REPLACE command, and only for a queued replacement from which VDE expects to generate a fold record.

User Action: You may/will need to resolve the fold record that will (eventually) be generated by this replacement. Use the /NOLOG qualifier for the command if you do not wish to see the message.

FOLDIDREQ, fold identifier required for module *module*

Facility: VDE, OpenVMS Development Environment

Explanation: You have multiple fold records for the indicated module. You must therefore specify a fold identifier with the /IDENTIFICATION or /FOLD qualifier (depending on the command) to uniquely indicate to which fold record you want the command to apply.

User Action: Reissue the command with the /IDENTIFICATION or /FOLD qualifier to indicate the specific fold record to which you want the command to apply.

FOLDIDUSED, fold identifier *identifier* already used for module *module*
attempt to create fold record has failed

Facility: VDE, OpenVMS Development Environment

Explanation: You are trying to create a fold record for a module using the same fold identifier as you have used on a previous fold record for the same module in the same development stream. This is not allowed, since fold identifiers must be unique.

User Action: Reissue the CREATE FOLD command with a different fold identifier specified on the /IDENTIFICATION qualifier. Alternatively, you can omit the /IDENTIFICATION qualifier, in which case VDE will assign a unique identifier for you.

FOLDINSTRM, generation *generation* must be folded into stream *stream*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that you must manually fold the indicated module generation into the indicated development stream in order to propagate the changes in that generation into the stream. VDE cannot propagate the changes automatically because the module has already diverged between that stream and its predecessor. This message

is only issued when you use the /LOG qualifier on the REPLACE or PERFORM REPLACEMENT command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

FOLDMOD, fold record *record* for module *module* modified in database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that fold record for the indicated module has been modified in the VDE database. This message is only issued when you use the /LOG qualifier on the MODIFY FOLD command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

FOLDNOTFOUND, fold record *record* not found for module *module* and stream *stream*

Facility: VDE, OpenVMS Development Environment

Explanation: You have tried to reference a fold record for the indicated module in the indicated development stream, but there is no such fold record.

User Action: This is not necessarily an error; the message simply tells you that the indicated module has no fold records in the indicated stream.

FROMDIREMPTY, the directory *directory* specified with /FROM_DIRECTORY is empty

Facility: VDE, OpenVMS Development Environment

Explanation: Since the directory specified with the /FROM_DIRECTORY qualifier is empty, no modules are created.

User Action: None.

GBLSECNAM, communications global section name: *gblsecnam*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you of the name of the communications section used within VDE. This message is associated with a more serious error of some sort. Resolve the basic error first.

User Action: If unable to resolve the basic error, please contact VDE support for assistance. Please include any information that may be useful or relevent in duplicating the problem.

GENDEL, generation *generation* deleted from stream *stream*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated generation has been deleted from the indicated development stream. This message is only

issued when you use the /LOG qualifier on the DELETE GENERATION command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

GENDELLIB, generation *generation* deleted from library

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated generation has been deleted from the library. This message is only issued when you use the /LOG qualifier on the DELETE GENERATION command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

GENINSERT, generation *generation* inserted into stream *stream*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated generation has been inserted into the indicated development stream. This message is only issued when you use the /LOG qualifier on the CREATE MODULE or INSERT GENERATION command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

GENMOD, information for generation *generation* modified in the database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that information for the indicated generation of the indicated module has been modified in the VDE database. This message is only issued when you use the /LOG qualifier on the MODIFY GENERATION command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

GENNODEL, generation *generation* cannot be deleted

Facility: VDE, OpenVMS Development Environment

Explanation: You are trying to delete a generation of a module which is still reachable by at least one stream and/or is not a leaf generation (some other generation points to it as its previous generation). VDE does not allow a generation to be deleted in either of these cases.

User Action: Check that you have specified the correct generation for deletion.

GENNOPRED, generation *generation* of *module* has no predecessor

Facility: VDE, OpenVMS Development Environment

Explanation: The generation of the indicated module with the shown CMS generation expression does not have a predecessor generation. As a result, the current operation cannot be performed. For example, the DIFFERENCES command cannot compare generation 1 with its predecessor because it has no predecessor.

User Action: Specify a generation that has a predecessor (normally any generation except generation 1) and rerun the command.

GENNOTCLASSMEM, generation *generation* is not a member of CMS class *class*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated generation is not a member of the indicated CMS class. This message is output when a VDE stream is maintaining CMS class membership, but the indicated generation is not a member of that CMS class.

User Action: Use the VERIFY GENERATION/RECOVER to attempt recovery of the class memebership for this stream.

GENNOTFOUNDCMS, generation *generation* not found in CMS library

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated module generation does not have a generation in the CMS library.

User Action: Use the VERIFY GENERATION/RECOVER command to attempt recovery of the missing generation or locate missing generation from a backup and manually insert the generation.

GENNOTINSTAGE, generation not found in staging area *area*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the module generation is not in the library staging area

User Action: Try to locate a previous module generation by checking with users that last had the module reserved.

GENNOTQUEREPL, replacement was not queued; no staging area exists

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the module generation has not been queued for replacement and therefore a generation cannot be located for recovery.

User Action: Try to locate a previous module generation by checking with users that last had the module reserved.

GENNOTRECOVERED, generation *generation* not recovered

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the module generation has not been recovered. The message that follows will provide more detail as to why the recovery failed.

User Action: Review the cause of the recovery failure. Try to locate a previous module generation.

GENRECCLASSMEM, generation *generation* successfully recovered into CMS class *class*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the module generation has been successfully recovered into the indicated CMS class.

User Action: None required.

GENRECOVERED, generation *generation* successfully recovered from *source*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the module generation has been successfully recovered.

User Action: None required.

GENREM, generation *generation* removed from stream *stream*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated generation has been removed from the indicated development stream. This message is only issued when you use the /LOG qualifier on the REMOVE GENERATION command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

GENREPLDNF, staging directory *directory* not found

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that reported staging directory does not exist. Perhaps the directory was deleted as part of a cleanup operation.

User Action: Try to locate a previous module generation by checking with users that last had the module reserved.

GENRETAINED, generation *generation* retained in library

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated generation has been retained in the library because it is still reachable by at least one

stream. This message is only issued when you use the /LOG qualifier on the DELETE GENERATION command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

GENSUCVERIFY, generation *generation* successfully verified

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated module generation has a corresponding generation in the CMS library and it is a member of all the CMS classes that are being maintained for the corresponding VDE streams.

User Action: None required.

GETJPI, SYS$GETJPI system service failed, unable to attach

Facility: VDE, OpenVMS Development Environment

Explanation: The ATTACH command could not continue because a failed service call to SYS$GETJPI occurred.

User Action: Determine if there is any external interference. If no interference can be found, then please submit a problem report.

GETQUIERR, get queue information system service error for queue *queue*

Facility: VDE, OpenVMS Development Environment

Explanation: You have attempted to start a build job and the VMS get queue information ($GETQUI) system service returned an error status. The message that follows gives more information to explain the error.

User Action: The action to take to correct the error depends on the nature of the error. You or your system manager should correct the error and restart the build job.

GRPADD, module *module* is added to group *group*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated module has been added as a member of the indicated group . This message is only issued when you use the /LOG qualifier on the CREATE GROUP or MODIFY GROUP command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

GRPADDED, group *group* added to the database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated group definition has been added to the VDE database. This message is only

issued when you use the /LOG qualifier on the CREATE GROUP command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

GRPALRMEM, module *module* is already a member of group *group*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that you have attempted to add the indicated module as a member of the indicated group, but the module is already a member of that group. This message is only issued when you use the /LOG qualifier on the MODIFY GROUP command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

GRPDEL, group *group* deleted from the database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the definition for the indicated group has been deleted from the VDE database. This message is only issued when you use the /LOG qualifier on the DELETE GROUP command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

GRPEXISTS, group *group* already exists in database; group not changed

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated group name was specified on a CREATE GROUP command, but an entry for that group is already present in the database. Hence a new entry was not created for that facility. Alternatively, the indicated name was specified as the new name for an existing group on the MODIFY GROUP command, but another group already has the requested new name. Hence the name of the existing group could not be changed to the new name.

User Action: If you entered the wrong group name, reenter the command with the correct name. If you entered the right group name, use the SHOW GROUP command to display the database information for that group. If you are not satisfied with that information, use the MODIFY GROUP command to change it.

GRPMOD, information for group *group* modified in the database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that information for the indicated group has been modified in the VDE database. This message is only issued when you use the /LOG qualifier on the MODIFY GROUP command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

GRPNOTFOUND, group *group* not found in database; parameter ignored

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated group name was entered on a SHOW GROUP command, but no such group was found in the VDE database.

User Action: This is not necessarily an error; the message simply tells you that the indicated group does not exist in the database.

GRPNOTMEM, module *module* is not a member of group *group*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that you have tried to remove the indicated module from the indicated group, but the module is not a member of that group. This message is only issued when you use the /LOG qualifier on the MODIFY GROUP command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

GRPREM, module *module* is removed from group *group*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated module has been removed from being a member of the indicated build set. This message is only issued when you use the /LOG qualifier on the MODIFY GROUP command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

GRPSYNTAX, invalid group name syntax: *name*

Facility: VDE, OpenVMS Development Environment

Explanation: The name or format specified for the group name is not using valid syntax.

User Action: Respecify the group name using only alphanumeric, underscore, dollar sign and hyphen characters.

HELLO, Motif *vdeversion* starting up

Facility: VDE, OpenVMS Development Environment

Explanation: This message displays the version identification of the VDE Motif interface in use. This message is displayed automatically during the VDE Motif startup.

User Action: None.

ICIBOGUS, intermediate conference information invalid

Facility: VDE, OpenVMS Development Environment

Explanation: This message is displayed in conjunction with other error messages, and indicates the particular failure that occurred.

User Action: Resolve the basic error. If unable to resolve the error, please contact VDE support for assistance.

ICIGONE, intermediate conference information not found

Facility: VDE, OpenVMS Development Environment

Explanation: This message is displayed in conjunction with other error messages, and indicates the particular failure that occurred.

User Action: Resolve the basic error. If unable to resolve the error, please contact VDE support for assistance.

ICIPARSE, parse of intermediate conference information failed

Facility: VDE, OpenVMS Development Environment

Explanation: This message is displayed in conjunction with other error messages, and indicates the particular failure that occurred.

User Action: Resolve the basic error. If unable to resolve the error, please contact VDE support for assistance.

ILLDEPFUN, Illegal dependency function passed to module VDE$REC_ DEPENDENCY. The illegal dependency function value is *value*

Facility: VDE, OpenVMS Development Environment

Explanation: An illegal dependency function was passed to the dependency recording module VDE$REC_DEPENDENCY. Valid dependency recording functions are defined in the module VDELIB.REQ.

User Action: Check the dependency recording function that was passed on the call to VDE$REC_DEPENDENCY. If that is ok, check the values assigned to the VDE$K_DEPEND_xxxx dependency recording literals in VDELIB.REQ. Someone may have assigned them incorrect values.

ILLFORUSE, ill-formed user defined name

Facility: VDE, OpenVMS Development Environment

Explanation: In the DEFINE/COMMAND command, the provided user-defined name is illegal.

User Action: Reissue the command with the correct user defined name.

INCNOTVAL, INPUT not valid on streams marked for queued replacement

Facility: VDE, OpenVMS Development Environment

Explanation: The specified or default stream is marked for queued replacements, and the CREATE MODULE /INPUT command cannot be

processed as it attempts to directly create the module(s) without regard to the review and replacement policy associated with the stream.

User Action: Perform a CREATE MODULE sans the /INPUT qualifier, and then use a REPLACE command. [This restriction may be relaxed in a future release of VDE.]

INMODNOTFOU, input module *module* of build step not found in database; ignored

Facility: VDE, OpenVMS Development Environment

Explanation: The automatically collected dependency information for the current build step specifies that the indicated module is an input to the step. However, the module is not found in the VDE database; it is neither specified as a source module nor as an output of any previous step. This indicates that the VDE database is not consistent with the actual contents of the facility subdirectory in which the compiler or other processor found the module.

User Action: Determine whether the module should be in the VDE library. If it should be there, create the module with the CREATE MODULE command. If it should not be there, delete it from the facility subdirectory in which the compiler or other processor found it.

INPNOTNAME, input string *text* is not syntactically valid

Facility: VDE, OpenVMS Development Environment

Explanation: An input string entered in response to an interactive query for the name of some entity (such as a library, module, or group) does not have the correct syntactic form for that kind of entity name.

User Action: Reenter a correctly specified name in response to the query.

INPTOOLONG, input string too long; maximum length is *number* characters

Facility: VDE, OpenVMS Development Environment

Explanation: An input string entered in response to an interactive query for information is too long because it exceeds the maximum length indicated in the message.

User Action: Reenter a shorter input string in response to the query.

INPTOOSHORT, input string too short; minimum length is *number* characters

Facility: VDE, OpenVMS Development Environment

Explanation: An input string entered in response to an interactive query for information is too short because it does not reach the minimum length indicated in the message.

User Action: Reenter a longer input string in response to the query.

INTERR, internal VDE error in *VDE-internal-identifier*

Facility: VDE, OpenVMS Development Environment

Explanation: VDE has detected an internal inconsistency.

User Action: Please contact VDE support for assistance. Please include any information that may be useful or relevent in duplicating the problem.

INTERR_AZ, internal VDE error in *VDE-internal-identifier*

Facility: VDE, OpenVMS Development Environment

Explanation: VDE has detected an internal inconsistency.

User Action: Please contact VDE support for assistance. Please include any information that may be useful or relevent in duplicating the problem.

INTFUNCERR, internal script function error

Facility: VDE, OpenVMS Development Environment

Explanation: The script function encountered an error during the parsing, and cannot be processed.

User Action: Please contact VDE support for assistance. Please include any information that may be useful or relevent in duplicating the problem.

INTMEMERR, internal memory-pool error at *hex-virtual-address VDE-internal-identifier*

Facility: VDE, OpenVMS Development Environment

Explanation: VDE has detected an inconsistency in its internal memory pool.

User Action: Please contact VDE support for assistance. Please include any information that may be useful or relevent in duplicating the problem.

INTTOOBIG, integer too large

Facility: VDE, OpenVMS Development Environment

Explanation: Integer values entered from the command line must fit into a longword.

User Action: Determine if it is necessary to provide such a large number. If it is necessary, then this is a restriction of VDE.

INVALANS, *answer* is an invalid answer

Facility: VDE, OpenVMS Development Environment

Explanation: The response to a confirmation message is invalid; only the word YES or the word NO or an abbreviation of one of those two words is allowed as a response. Confirmation messages only appear when you use the /CONFIRM qualifier on a command that accepts this qualifier.

User Action: Reenter either YES or NO (or Y or N) in response to the confirmation message.

INVDATLNM, invalid VDE$DATE_TIME; use DD-MMM-YYYY:HH:MM:SS.CC format: *time*

Facility: VDE, OpenVMS Development Environment

Explanation: The format of a date-time value entered as a logical name is invalid. This message should appear only during VDE testing, and is only displayed by debug versions of VDE, and the logical name is only translated by debug versions.

User Action: Contact VDE support for assistance, as the version of VDE in use contains debug-only images. If the logical name is being specified, correct the error in the date-time logical name value and reissue the command.

INVDATTIM, invalid date-time; use DD-MMM-YYYY:HH:MM:SS.CC format: *time*

Facility: VDE, OpenVMS Development Environment

Explanation: The format of a date-time value entered as a command parameter is invalid.

User Action: Correct the error in the date-time value and reissue the command.

INVDBFORM, invalid database format; VDE is unable to access library

Facility: VDE, OpenVMS Development Environment

Explanation: The VDE database does not have the proper format. The database header record could be read successfully, but that record does not contain the proper sentinel value or its database version value indicates an out-of-date or invalid database format. This error could indicate that the database has been corrupted in some way or that VDE has generated it incorrectly.

User Action: Try to connect to the database again using the SET LIBRARY command. If that does not work, you may have to recreate the library (and thus the database) from scratch with the CREATE LIBRARY command. If that still does not work, please contact VDE support for assistance.

INVDIGDEC, invalid decimal number: *number*

Facility: VDE, OpenVMS Development Environment

Explanation: A decimal digit is often expected in many contexts such as a parameter. If the provided input is not a digit, this error message will be generated.

User Action: Reissue the command with the appropriate number.

INVDIRSPEC, invalid directory specification: *text*

Facility: VDE, OpenVMS Development Environment

Explanation: A directory specification on a VDE command is not syntactically correct. This message is also printed if a logical name (such as VDE$LIBRARY) that is supposed to be bound to a directory specification is not bound to a syntactically correct directory specification.

User Action: Reenter the VDE command with a valid directory specification or define the logical name to be bound to a valid directory specification.

INVGENEXPR, invalid CMS generation expression: *expression*

Facility: VDE, OpenVMS Development Environment

Explanation: A CMS generation expression on a VDE command is not syntactically correct.

User Action: Reenter the VDE command with a valid CMS generation expression.

INVHISCHAR, invalid history or notes character following poundsigninvalid string is: *string*

Facility: VDE, OpenVMS Development Environment

Explanation: A history or notes string was specified which contains an invalid character after the poundsign character. Valid characters for a history string are: H, h, B, or b. Valid characters for a notes string are: G or g. A poundsign character may be included in the string by quoting it with another poundsign character.

User Action: Reenter the VDE command with a valid history or notes string.

INVINPUT, invalid interactive command input; command aborted

Facility: VDE, OpenVMS Development Environment

Explanation: The current command has queried the user for input and has gotten invalid input several times in a row. This can be caused by specifying insufficient parameter or qualifier information on a command in a command file, which may cause subsequent commands to be read as responses to interactive queries. Due to the invalid input, the current command is aborted.

User Action: Reissue the command with correct qualifiers and parameters or with correct input to interactive queries for command information.

INVINTVAL, invalid integer value on command line: *integer*

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated integer value was entered on the current command line but its value is not in the valid range for that kind of value. For example, if the number of seconds in a delta-time interval is exceeds 59, you will get this error message.

User Action: Reissue the command with a valid integer value.

INVIPMTNAM, invalid IPMT/CLD number syntax

Facility: VDE, OpenVMS Development Environment

Explanation: The format for the CLD number is not valid.

User Action: Respecify the CLD number in the format ABC-12345, ABC-123456, ABC.12345 or ABC.123456. If the IPMT/CLD number is not in this format, or if this format is not accepted by VDE, please contact VDE support for assistance.

INVKEYNAM, invalid or missing key name

Facility: VDE, OpenVMS Development Environment

Explanation: The name of a key on the keyboard was expected in this command.

User Action: Reissue the command with an acceptable name of a key.

INVNUMRNG, number not valid; number is *number*, valid range is *number* to *number*

Facility: VDE, OpenVMS Development Environment

Explanation: A numeric input value is not within its valid range. The valid range is indicated in the message.

User Action: Reenter the number with a value within the valid range.

INVNUMSTG, invalid numeric string: *string*

Facility: VDE, OpenVMS Development Environment

Explanation: A numeric input value is not syntactically valid.

User Action: Reenter the number with correct syntax.

INVPAGE, invalid screen height, value must be between *minimum_height* and *maximum_height*

Facility: VDE, OpenVMS Development Environment

Explanation: For the SET TERMINAL command, the /PAGE qualifier must provide a page length between 11 and 100, inclusively.

User Action: Reissue the SET TERMINAL command with an acceptable page size.

INVPARAM, invalid parameter passed to *routine-name-plus-info*

Facility: VDE, OpenVMS Development Environment

Explanation: VDE has detected in internal inconsistency, namely that an illegal value has been passed as a parameter to an internal routine.

User Action: Please contact VDE support for assistance. Please include any information that may be useful or relevent in duplicating the problem.

INVQARDB, invalid QAR database name syntax

Facility: VDE, OpenVMS Development Environment

Explanation: The format for the QAR database name is not valid.

User Action: Respecify the QAR database name using only alphanumeric, underscore, dollar sign or hyphen characters.

INVQARNO, invalid QAR entry number syntax

Facility: VDE, OpenVMS Development Environment

Explanation: The format of the QAR entry number is not valid.

User Action: Respecify the QAR entry number using only numeric characters.

INVRANGE, no values in inverted range *number* to *number*

Facility: VDE, OpenVMS Development Environment

Explanation: The values specified for the /IDENTIFICATION qualifier on the DELETE, MODIFY or SHOW REPLACEMENT command are in reverse order and, therefore, do not represent any value. For this reason, no replacement entries are found matching this criteria.

User Action: Reenter the command specifying a valid range of replacement id numbers in ascending order.

INVSOURCE, Invalid source of information specified: *code*

Facility: VDE, OpenVMS Development Environment

Explanation: The source of information—typically the SCT, FOLD or OTHER—is not valid.

User Action: Reenter a valid source of information on this change.

INVWIDTH, invalid screen width, value must be between *minimum_width* and *maximum_width*

Facility: VDE, OpenVMS Development Environment

Explanation: For the SET TERMINAL command, the /WIDTH qualifier can provide a page length between 40 and 255, inclusively.

User Action: Reissue the SET TERMINAL command with an acceptable width.

INVWILDSPEC, invalid wildcard specification in script selector: *selector*

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated module name or facility name is wildcarded in a way that is not valid in a module-name or facility specification used to select scripts. The only form of wildcarding that is allowed in this context is a single asterisk (*); other, more general forms of wildcarding are not allowed in script selectors. Also, the components of a full module (such as [FAC]MOD.TYP) may all be wildcarded individually, but only in certain combinations. Invalid combinations yield this error message.

User Action: Reissue the command with an acceptable wildcard specification.

IPMTAGAIN, please re-enter the IPMT or CLD number

Facility: VDE, OpenVMS Development Environment

Explanation: There is a problem with the IPMT or CLD name, and you are being prompted to re-enter the IPMT or CLD number. This message should be paired with other error messages, and these other messages should provide insight into the particular problem VDE is (currently) griping about.

User Action: The the IPMT or CLD name should be specified using three leading alphabetic characters, a hyphen or period, and five or six trailing numeric characters. ABC-12345, ABC.12345, ABC.123456, and ABC-123456 are all examples of valid IPMT (or CLD) name syntax.

IPMTFMT, use ABC.12345, ABC-12345, ABC.123456 or ABC-123456

Facility: VDE, OpenVMS Development Environment

Explanation: These are valid formats for the IPMT/CLD number.

User Action: The IPMT/CLD number must be specified in the format ABC-12345, ABC-123456, ABC.12345 or ABC.123456. If the IPMT/CLD number is not in this format, or if this format is not accepted by VDE, please contact VDE support for assistance.

IPMTSYNTAX, invalid IPMT/CLD name syntax: *name*

Facility: VDE, OpenVMS Development Environment

Explanation: The name or format specified for the IPMT/CLD name is not using valid syntax.

User Action: Respecify the IPMT/CLD name using three leading alphabetic characters, a hyphen or period, and five or six trailing numeric characters. ABC-12345, ABC.12345, ABC.123456, and ABC-123456 are all examples of valid IPMT/CLD name syntax.

ITEMLIST, unable to parse itemlist list entry at offset *offset*

Facility: VDE, OpenVMS Development Environment

Explanation: VDE has detected a size, type code or buffer address error at the specified entry offset in the itemlist.

User Action: Please evaluate the specified entry offset in the itemlist, and rework it as appropriate.

IVQLOC, invalid qualifier location - place after each dependency file parameter

Facility: VDE, OpenVMS Development Environment

Explanation: The /INPUT and /OUTPUT qualifiers are allowed as a qualifier for each file specification that is to be recorded as a dependency. These qualifiers are only meaningful for the build dependency parameter of the VDEDEPEND command.

User Action: Reissue the command with the proper syntax.

IVQUAL, unrecognized qualifier '*qualifier*' - check validity, spelling, and placement

Facility: VDE, OpenVMS Development Environment

Explanation: An undefined qualifier was found.

User Action: Verify the syntax for the command and reissue it with only the allowed qualifiers.

JOURDISA, database journalling disabled for VDE library

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that after-image journalling has been disabled (turned off) for the VDE library. When journalling is disabled, it can be very hard to recover from a damaged or corrupted VDE database.

User Action: None.

JOURENA, database journalling enabled for VDE library

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that after-image journalling has been enabled (turned on) for the VDE library. Journalling makes it possible to recover the database contents if the database is damaged or corrupted.

User Action: None.

KEYADDED, keyword *keyname* added to note *noteID* in conference *conference*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that VDE has successfully added the indicated keyword to a note in the indicated notes conference.

User Action: None

LIBCONV, library format converted for new version of VDE

Facility: VDE, OpenVMS Development Environment

Explanation: The format of the current VDE library, including its database schema, has been converted to the format required by the present version of VDE. If automatic library conversion is enabled for the current VDE library, this action takes place the first time someone accesses the VDE database with a new version of VDE. If automatic library conversion is disabled, you must enter the CONVERT LIBRARY command to convert the library.

User Action: None. This is an informational message only.

LIBINDRAN, Library index value of *value* out of range for library dependency or module dependency.

Facility: VDE, OpenVMS Development Environment

Explanation: A library dependency or module dependency with an illegal library index value was passed to the module VDE$REC_DEPENDENCY for dependency recording. The value was equal to 0, or greater than the currently largest assigned library index value for the dependency recording session.

User Action: Check the call to LIB$REC_DEPENDENCY for recording this library dependency or module dependency for a correct library index value. If the library index value appears correct please submit a problem report.

LIBLABEL, library (*mnemonic*) contents: *label*

Facility: VDE, OpenVMS Development Environment

Explanation: This is an informational message, and indicates the label associated with a particular library that is available.

User Action: No action is required.

LIBMOD, library attributes modified in database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that attributes for the current library have been modified in the VDE database. This message is only issued when you use the /LOG qualifier on the MODIFY LIBRARY command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

LIBNOTSET, library not set; no VDE database found in directory *directory*

Facility: VDE, OpenVMS Development Environment

Explanation: You have attempted to set a VDE library by specifying a directory specification or library mnemonic specification on the SET LIBRARY command, but the directory or mnemonic either does not exist or does not reference a valid VDE library database. If you are using a mnemonic for the library, check to make sure you have entered a valid

mnemonic, then check to see that the VDE$LIBRARY_DEFAULTS* logical names all include the appropriate and matching numbers of translations. (If you are uncertain how to check the VDE$LIBRARY_DEFAULTS* logical names, consult the VDE user's guide.)

User Action: Reenter the command with a directory specification that specifies a valid root directory for a VDE library, or enter a correct mnemonic. If the directory or mnemonic is correct, contact the local system management staff and ask that the translations of the VDE$LIBRARY_ DEFAULTS* logical name(s) be corrected.

LIBREQ, specification of library mnemonic required

Facility: VDE, OpenVMS Development Environment

Explanation: VDE/LOCATIONS is unable to locate the target for the location translation operation.

User Action: Specify the target via VDE/LOCATIONS/MNEMONIC=mnemonic.

LIBRESET, library reset

Facility: VDE, OpenVMS Development Environment

Explanation: You have reset the VDE library.

User Action: No action is required.

LINKFAIL, link of *module* completed with errors

Facility: VDE, OpenVMS Development Environment

Explanation: The link step for the indicated module completed with one or more errors. The linker's output (on your terminal or in the step's log file) gives more detail on the errors it detected.

User Action: Correct the errors and resubmit the link or build job.

LINKSUCC, link of *module* completed successfully

Facility: VDE, OpenVMS Development Environment

Explanation: The link of the indicated module completed successfully.

User Action: None.

LNKSTEPADDED, link step for *module* added to database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated link step has been added to the VDE database. This message is only issued when you use the /LOG qualifier on the CREATE STEP command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

LNKSTEPDEL, link step for module *module* deleted from database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the link step for the indicated module has been deleted from the VDE database. This means that all build dependency information for that build step has been deleted. This message is only issued when you use the /LOG qualifier on the DELETE STEP command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

LNKSTEPMOD, link step for module *module* modified in database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that build dependency information for the indicated link step has been modified in the VDE database. This message is only issued when you use the /LOG qualifier on the MODIFY STEP command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

LOASTLM, recommended free ASTLM value is *value*

Facility: VDE, OpenVMS Development Environment

Explanation: The amount of the ASTLM quota currently available is below the recommended value for VDE. VDE will attempt to continue.

User Action: Contact your system management staff and request that the process quotas associated with the current username be reset to the values recommended for use with VDE. This message includes the recommended minimum free value for the quota.

LOBIOLM, recommended free BIOLM value is *value*

Facility: VDE, OpenVMS Development Environment

Explanation: The amount of the BIOLM quota currently available is below the recommended value for VDE. VDE will attempt to continue.

User Action: Contact your system management staff and request that the process quotas associated with the current username be reset to the values recommended for use with VDE. This message includes the recommended minimum free value for the quota.

LOBYTLM, recommended free BYTLM value is *value*

Facility: VDE, OpenVMS Development Environment

Explanation: The amount of the BYTLM quota currently available is below the recommended value for VDE. VDE will attempt to continue.

User Action: Contact your system management staff and request that the process quotas associated with the current username be reset to the values recommended for use with VDE. This message includes the recommended minimum free value for the quota.

LOCKED, library write access is disabled

Facility: VDE, OpenVMS Development Environment

Explanation: This message is displayed when an attempt is made to write—RESERVE, REPLACE or PERFORM REPLACE a module, etc—into to a locked library.

User Action: Contact the staff responsible for the library to determine why access to the library is currently locked out. The library may be currently undergoing maintenance operations, or the library may be a read-only shadow copy of a library. Only a user with the MODLIB privilege can unlock the library.

LODIOLM, recommended free DIOLM value is *value*

Facility: VDE, OpenVMS Development Environment

Explanation: The amount of the DIOLM quota currently available is below the recommended value for VDE. VDE will attempt to continue.

User Action: Contact your system management staff and request that the process quotas associated with the current username be reset to the values recommended for use with VDE. This message includes the recommended minimum free value for the quota.

LOENQLM, recommended free ENQLM value is *value*

Facility: VDE, OpenVMS Development Environment

Explanation: The amount of the ENQLM quota currently available is below the recommended value for VDE. VDE will attempt to continue.

User Action: Contact your system management staff and request that the process quotas associated with the current username be reset to the values recommended for use with VDE. This message includes the recommended minimum free value for the quota.

LOFILLM, recommended free FILLM value is *value*

Facility: VDE, OpenVMS Development Environment

Explanation: The amount of the FILLM quota currently available is below the recommended value for VDE. VDE will attempt to continue.

User Action: Contact your system management staff and request that the process quotas associated with the current username be reset to the values recommended for use with VDE. This message includes the recommended minimum free value for the quota.

LOGNAMUNDEF, logical name *logical* undefined; operation cannot complete

Facility: VDE, OpenVMS Development Environment

Explanation: The current VDE operation requires the indicated logical name to be defined, but it is not defined. As a result, VDE cannot perform the operation.

User Action: Exit the VDE kept process with the EXIT/KEPT_PROCESS command, define the required logical name, restart VDE, and retry the operation.

LOJTQUOTA, recommended free JTQUOTA is *value*

Facility: VDE, OpenVMS Development Environment

Explanation: The amount of the JTQUOTA currently available is below the recommended value for VDE. VDE will attempt to continue.

User Action: Contact your system management staff and request that the process quotas associated with the current username be reset to the values recommended for use with VDE. This message includes the recommended minimum free value for the quota.

LOPGFLQUOTA, recommended free PGFLQUOTA value is *value*

Facility: VDE, OpenVMS Development Environment

Explanation: The amount of the PGFLQUOTA quota currently available is below the recommended value for VDE. VDE will attempt to continue.

User Action: Contact your system management staff and request that the process quotas associated with the current username be reset to the values recommended for use with VDE. This message includes the recommended minimum free value for the quota.

LOPRCLM, recommended free PRCLM value is *value*

Facility: VDE, OpenVMS Development Environment

Explanation: The amount of the PRCLM quota currently available is below the recommended value for VDE. VDE will attempt to continue.

User Action: Contact your system management staff and request that the process quotas associated with the current username be reset to the values recommended for use with VDE. This message includes the recommended minimum free value for the quota.

LOQUOTA, one or more process quotas is too low

Facility: VDE, OpenVMS Development Environment

Explanation: The currently-free value of one or more quotas is below the recommended values for VDE. VDE will attempt to continue.

User Action: The determination of a low quota is based on the currently-free value of the quota. Contact your system management staff and request that the process quotas associated with the current username be reset to provide the recommended value with VDE. One or more messages are output with this message listing the particular quota found to be low, and the recommended minimum free value for the quota. VDE will attempt to continue from this error.

LOTQELM, recommended free TQELM value is *value*

Facility: VDE, OpenVMS Development Environment

Explanation: The amount of the TQELM quota currently available is below the recommended value for VDE. VDE will attempt to continue.

User Action: Contact your system management staff and request that the process quotas associated with the current username be reset to the values

recommended for use with VDE. This message includes the recommended minimum free value for the quota.

MAILERR, error from MAIL routine *routine-name*

Facility: VDE, OpenVMS Development Environment

Explanation: VDE has encountered an error while using the VMS MAIL callable interface.

User Action: Attempt to resolve the problem based on any other error messages included with this one and contact VDE support for assistance.

MAILSENT, mail message sent to user *user*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that VDE has sent a mail message to the indicated user notifying that user of your action. This message is issued when you use the /LOG qualifier on a command that sends mail notifications.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

MATQUOMIS, matching quote is missing

Facility: VDE, OpenVMS Development Environment

Explanation: All quoted strings must be delimited by either two single or two double quotes.

User Action: Reenter the command with appropriately quoted string.

MAXIDVAL, maximum allowed Id value in database exceeded: *ID*

Facility: VDE, OpenVMS Development Environment

Explanation: Each entity in the VDE database, such as each module, each stream, and each build job, has an integer Id value that uniquely identifies it. This Id value may not assume a value larger than 2,000,000,000, but the operation you have attempted to perform has tried to create an entity with an Id value larger than this maximum. The indicated name tells which Id value is affected. This error should never occur unless the database is extremely large.

User Action: Please contact VDE support for assistance. Most likely the VDE database has been damaged by an internal error.

MAXPARM, too many parameters - reenter command with fewer parameters

Facility: VDE, OpenVMS Development Environment

Explanation: In both the SHOW KEY and DELETE KEY commands, the user cannot provide both /ALL as a qualifier and a key name as a parameter.

User Action: Choose between /ALL or just one key in either the SHOW KEY or DELETE KEY commands.

MERGEGEN, merging generation *generation*(*this*) into *that*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the current FETCH or RESERVE command will merge the indicated ‘‘merge’’ generation into the generation being fetched or reserved to produce the output file. This message is only issued when you use the /MERGE and /LOG qualifiers on the command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

MISCONTSCRP, missing continuation line in script

Facility: VDE, OpenVMS Development Environment

Explanation: The current script specified that the last line should be continued but no such continuation line exists. A hyphen (-) at the end of a line indicates that the line should be continued.

User Action: Either remove the hyphen at the end of the last line or supply a continuation line, whichever is appropriate.

MISHISPARM, missing history parameter in string "*string*"

Facility: VDE, OpenVMS Development Environment

Explanation: A history string was specified which does not contain a history parameter. A history parameter is required as part of the history string.

User Action: Reenter the VDE command with a valid history string.

MISTYPNAM, missing type name for module *string*; type name required

Facility: VDE, OpenVMS Development Environment

Explanation: A module name has been specified without a module type in a context where the module type is required. For example, the command CREATE MODULE M1 is incorrect because a module type, as in CREATE MODULE M1.FOR (where FOR is the module type), is required for this command. The type name must be separated from the module name by a dot (.) as shown in the example.

MNEMONIC, library (*mnemonic*) *library* available

Facility: VDE, OpenVMS Development Environment

Explanation: This is an informational message, and indicates a particular library is available, as well as the mnemomic name associated with the library.

User Action: No action is required.

MNETOOLONG, library mnemonic too long; maximium length: *length*

Facility: VDE, OpenVMS Development Environment

Explanation: One (or more) of the VDE$LIBRARY_DEFAULTS_ MNEMONICS library mnemonic name translations are too long.

User Action: Contact the maintainer of the definitions of the VDE$LIBRARY_DEFAULTS_MNEMONICS logical name in use, and ask that the mnemonics be shortened.

MODADDED, module *module* added to the database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated module definition has been added to the VDE database. This message is only issued when you use the /LOG qualifier on the CREATE MODULE command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

MODALRINSTEP, module *module* is already an input to build step

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated module is already an input to the build step. This message is only issued when you use the /LOG qualifier on the MODIFY STEP command and specify a new input module which is already an input to the step.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

MODALROUTSTEP, module *module* is already an output from build step

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated module is already an output from the build step. This message is only issued when you use the /LOG qualifier on the MODIFY STEP command and specify a new output module which is already an output from the step.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

MODALRQUEREP, module *module* already queued for replacement *replacement*

Facility: VDE, OpenVMS Development Environment

Explanation: You have tried to replace or unreserve a module that already belongs to the indicated queued replacement. A reservation that already belongs to a queued replacement cannot be unreserved or queued for another replacement; the reservation remains until the original replacement is performed or cancelled. As a result, your attempt to replace or unreserve the module has failed.

User Action: If you want to remove the module reservation from its current queued replacement, you must cancel that replacement with the

CANCEL REPLACEMENT command. You can then unreserve the module or replace it as part of another replacement.

MODALRRES, module *module* already reserved; reservation has failed

Facility: VDE, OpenVMS Development Environment

Explanation: You have tried to reserve a module that is already reserved by another user (possibly yourself). That reservation does not allow concurrent reservations, or you have not requested a concurrent reservation. The reservation may be in the current default stream, or in an ancestor stream. As a result of this prior reservation, your attempt to reserve the module has failed.

User Action: Wait until the user who has reserved the module replaces it in the library. Then reserve it yourself. You can use the SHOW RESERVATION/FULL command to see who currently has the module reserved, and in what stream the module was originally reserved. Alternatively, you can use the /OVERRIDE=CONCURRENT qualifier to concurrently reserve the module, or the /QUEUE qualifier to queue a request for notification when the module becomes available.

MODDEL, module *module* deleted from the database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the definition for the indicated module has been deleted from the VDE database. This message is only issued when you use the /LOG qualifier on the DELETE MODULE command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

MODEXISTS, module *module* already exists in database; module not changed

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated module name was specified on a CREATE MODULE command, but an entry for that module is already present in the database. Hence a new entry was not created for that module. Alternatively, the indicated name was specified as the new name for an existing module on the MODIFY MODULE command, but another module in the same facility already has the requested new name. Hence the name of the existing module could not be changed to the new name.

User Action: If you entered the wrong module name, reenter the command with the correct name. If you entered the right module name, use the SHOW MODULE command to display the database information for that module. If you are not satisfied with that information, use the MODIFY MODULE command to change it.

MODINSTEP, module *module* is an input to the build step

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated module is recorded as an input to the build step being defined. This message is only issued when you use the /LOG qualifier on the CREATE STEP or MODIFY STEP command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

MODLOCKED, module *module* locked for replacement by another user

Facility: VDE, OpenVMS Development Environment

Explanation: The module you are trying to replace is currently being replaced by another user who is updating the same generation as you want to update. As a result, your replacement attempt failed.

User Action: Wait a minute or two and then try the operation again. If the other user deleted your concurrent reservation through his or her replacement, your second attempt will fail with a message to that effect. If the other user's replacement failed or did not conflict with yours, your new replacement attempt can succeed.

MODMODIF, information for module *module* modified in the database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that information for the indicated module has been modified in the VDE database. This message is only issued when you use the /LOG qualifier on the MODIFY MODULE command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

MODNOTDEL, module *module* not deleted

Facility: VDE, OpenVMS Development Environment

Explanation: You are attempting to delete a module that cannot be deleted. Additional messages are normally included with this message, and these messages indicate why the module cannot be deleted.

User Action: Resolve the problem(s) flagged by the other error messages, and reenter the DELETE MODULE command.

MODNOTFOUND, module *module* not found in database; parameter ignored

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated module name was entered on a SHOW MODULE command, but no such module was found in the VDE database.

User Action: This is not necessarily an error; the message simply tells you that the indicated module does not exist in the database.

MODNOTINSTEP, module *module* is not now an input to build step

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated module is not now recorded as an input to the build step whose definition you are modifying. Removing it from the build step is thus redundant. This message is only issued when you use the /LOG qualifier on the MODIFY STEP command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

MODNOTINSTRM, generation *generation* not in stream *stream*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that an attempt was made to update the specified stream with a new generation of the specified module, but the specified module is not part of the specified development stream.

User Action: Use the /NOPROPAGATE qualifier for the command or populate the specified stream with the specified generation and try the command again.

MODNOTOUTSTEP, module *module* is not now an output from build step

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated module is not now recorded as an output from the build step whose definition you are modifying. Removing it from the build step is thus redundant. This message is only issued when you use the /LOG qualifier on the MODIFY STEP command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

MODNOTRES, module *module* is not currently reserved

Facility: VDE, OpenVMS Development Environment

Explanation: You are trying to replace the indicated module but that module is not currently reserved in the current stream development stream.

User Action: Reserve the module before trying to replace it.

MODNOTREV, module *module* not found in review list

Facility: VDE, OpenVMS Development Environment

Explanation: You have specified a module that does not exist in the current review list. As a result, you cannot select the module with the name you specified.

User Action: Reenter the SELECT command specifying a module that is present in the review list for the current REVIEW command. Use the DIRECTORY subcommand to determine what modules are present in the list.

MODOBS, module *module-name* superceded or marked obsolete

Facility: VDE, OpenVMS Development Environment

Explanation: The latest generation of the specified module indicates the module was superceded by a more recent module or was marked obsolete.

User Action: Determine if the current operation should proceed with a module that has been superceded or is marked as obsolete. If the operation should proceed, determine if any modifications are needed to any module(s) that may have superceded or replaced this module.

MODOUTSTEP, module *module* is an output from the build step

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated module is recorded as an output from the build step being defined. This message is only issued when you use the /LOG qualifier on the CREATE STEP or MODIFY STEP command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

MODREMINSTEP, module *module* removed as an input to build step

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated module is no longer recorded as an input to the build step whose definition you are modifying. This message is only issued when you use the /LOG qualifier on the MODIFY STEP command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

MODREMOUTSTEP, module *module* removed as an output from build step

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated module is no longer recorded as an output from the build step whose definition you are modifying. This message is only issued when you use the /LOG qualifier on the MODIFY STEP command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

MUSTREPL, you must replace the new module to create its first generation

Facility: VDE, OpenVMS Development Environment

Explanation: This message is displayed when you create a new module with the CREATE MODULE command and the command marks the module as reserved. It reminds you that you must replace the module with a REPLACE command to create the first generation of the module. This message is only issued when you use the /LOG qualifier on the CREATE MODULE command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

NAMTOOLONG, name is too long: *text*; maximum length is *number*

Facility: VDE, OpenVMS Development Environment

Explanation: A name entered as a command parameter is too long because it exceeds the maximum length indicated in the message.

User Action: Reissue the command with a shorter name.

NEEDFACILITY, this request requires a facility

Facility: VDE, OpenVMS Development Environment

Explanation: VDE was unable to perform the specified request as a facility was not selected nor otherwise specified.

User Action: Select or specify a facility, and retry the request.

NEEDLIBRARY, this request requires a library

Facility: VDE, OpenVMS Development Environment

Explanation: VDE was unable to perform the specified request as a library was not selected nor otherwise specified.

User Action: Select or specify a library, and retry the request.

NEEDMOD, module must be specified on command

Facility: VDE, OpenVMS Development Environment

Explanation: You have failed to specify a module name on a command that requires it.

User Action: Enter the module name you want on the command.

NEEDMODULE, this request requires a module

Facility: VDE, OpenVMS Development Environment

Explanation: VDE was unable to perform the specified request as a module was not selected nor otherwise specified.

User Action: Select or specify a module, and retry the request.

NEEDMORE, unexpected end of command line

Facility: VDE, OpenVMS Development Environment

Explanation: Many commands require the presence of additional parameters or qualifiers.

User Action: Review the correct syntax of the respective command and enter it with the appropriate parameters.

NEEDSTEP, build step must be specified on command

Facility: VDE, OpenVMS Development Environment

Explanation: You have failed to specify a build step on a command that requires it.

User Action: Enter a module name on the command to select the build step you want.

NEEDSTREAM, this request requires a stream

Facility: VDE, OpenVMS Development Environment

Explanation: VDE was unable to perform the specified request as a stream was not selected nor otherwise specified.

User Action: Select or specify a stream, and retry the request.

NETERROR, network error; component: *id*, code: *errcod*

Facility: VDE, OpenVMS Development Environment

Explanation: Something deep within the bowels of the VDE network layer has returned an error.

User Action: Please check the status of the local network, and then contact VDE support for further assistance.

NEWDBGMSK, debug mask switching from *oldmask* to *newmask*

Facility: VDE, OpenVMS Development Environment

Explanation: This message indicates that the value loaded in the VDE debugging bitmask is being changed. This message should appear only during VDE testing.

User Action: Contact VDE support for assistance, as the version of VDE in use contains debug-only images.

NOACCLIB, no access to library allowed for user *username*

Facility: VDE, OpenVMS Development Environment

Explanation: You are not authorized to access the VDE library that logical name VDE$LIBRARY points to or that you are trying to establish with a SET LIBRARY command.

User Action: Correct the definition of logical name VDE$LIBRARY, specify the name of a library you are authorized to access on the SET LIBRARY command, or request that the administrator for the desired VDE library give you the right to access that library.

NOARCHSPEC, no architecture specified for operation

Facility: VDE, OpenVMS Development Environment

Explanation: You have entered a command that requires a default architecture to exist, but you have not established a default architecture. Usually, this means that the command you have entered contains a module name with an omitted architecture name and the architecture cannot be defaulted. As a result, the module cannot be looked up in any architecture.

User Action: Either establish a default archtecture with the SET ARCHITECTURE command and reenter the command, or reenter the command with an explicitly specified architecture name, as in ARCHNAM:[FACNAM]MOD.TYP.

NOATTACH, failed to attach to process PID !XL

Facility: VDE, OpenVMS Development Environment

Explanation: The ATTACH command has failed to attach to the requested process.

User Action: Verify that the requested process exists. Check spelling of the process name.

NOAUTOCONV, automatic conversion to current database format disabled

Facility: VDE, OpenVMS Development Environment

Explanation: The VDE database does not have the proper format, and the automatic conversion of the database has been disabled.

User Action: If use of the new version of VDE is intended, perform a full library backup using the database-provided backup tools, and then use the VDE CONVERT command to update the database format.

NOBASENOTE, no base note found for replacement *replacement_id*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is displayed in conjunction with other error messages, and indicates the particular failure that occurred. VDE has attempted to find a base note with either a keyword or a title containing the specified replacement ID and has failed.

User Action: Resolve the basic error. If unable to resolve the error, please contact the project leader or VDE support for assistance.

NOBLDJOB, no build jobs found for stream *stream*

Facility: VDE, OpenVMS Development Environment

Explanation: No build job is present in the database that belongs to the indicated stream and matches the selection criteria you specified on the SHOW BUILD_JOB or other BUILD_JOB command. The selection criteria you have specified may include the build-job identification number, the build-job status, or bounds on the build-job creation date and time.

User Action: This is not necessarily an error; the message simply tells you that the no build job exists for the stream and selection criteria you specified.

NOBLDSTEPS, build job contains no build steps; build job is not created

Facility: VDE, OpenVMS Development Environment

Explanation: You tried to create a build job with the BUILD command, but the build job turns out to contain no build steps. As a result, the build job is not created in the database. The usual cause of this error is that you specified a minimal build of modules that are already up-to-date with respect to their source modules.

User Action: If you intended to perform a minimal build, the build job is unnecessary. If you intended to perform a full build, specify the /FULL qualifier on the BUILD command.

NOBUILD, no BUILD privilege; cannot create build jobs in this library

Facility: VDE, OpenVMS Development Environment

Explanation: You attempted to use a command or qualifier that requires one or more VDE privileges, and you did not have sufficient privileges enabled. This message is normally displayed in conjunction with other error messages, usually a no-privilege error.

User Action: If you are authorized to use the privilege, use the VDE SET PRIVILEGES command to give yourself the necessary privilege. Then try the original operation again. If you are not authorized to use the qualifier, you should either request the necessary privilege from your project administrator or you should not use the qualifier.

NOCANRES, cannot cancel reservation to create first generation of *module*

Facility: VDE, OpenVMS Development Environment

Explanation: You have entered an UNRESERVE command to cancel the reservation that will create the initial generation of the indicated module. The reservation was created by a CREATE MODULE command and VDE does not allow you to cancel such a reservation.

User Action: If you no longer want the reservation that will create the first generation of the module, you must delete the module with the DELETE MODULE command. If you just want to change the attributes of the reservation, you must do so with the MODIFY RESERVATION command.

NOCAUSE, replacement cause not known; module not replaced

Facility: VDE, OpenVMS Development Environment

Explanation: VDE was not provided with the cause of the replacement being performed. The cause of a replacement—usually a Source Code Tracking (SCT) entry or a source code Fold—is tracked.

User Action: Select the appropriate replacement cause—Source Code Tracking (SCT), Fold, or Other—using the toggle buttons on the module replacement dialog box.

NOCLOSEQ, no closing quotation mark for a quoted string

Facility: VDE, OpenVMS Development Environment

Explanation: For the DEFINE/COMMAND command, the parameter that describes the command must be enclosed with either single or double quotes.

User Action: Reenter the DEFINE/COMMAND command with the appropriate quotes.

NOCONFNAM, no notes conference name specified in call to *routine-name*

Facility: VDE, OpenVMS Development Environment

Explanation: VDE has detected an inconsistency within an internal routine which uses the notes callable interface.

User Action: Please contact VDE support for assistance. Please include any information that may be useful or relevent in duplicating the problem.

NOCREARCH, no CREARCH privilege; cannot create or modify architectures

Facility: VDE, OpenVMS Development Environment

Explanation: You attempted to use a command or qualifier that requires one or more VDE privileges, and you did not have sufficient privileges enabled. This message is normally displayed in conjunction with other error messages, usually a no-privilege error.

User Action: If you are authorized to use the privilege, use the VDE SET PRIVILEGES command to give yourself the necessary privilege. Then try the original operation again. If you are not authorized to use the qualifier, you should either request the necessary privilege from your project administrator or you should not use the qualifier.

NOCRECTX, no CRECTX privilege; cannot create or delete contexts

Facility: VDE, OpenVMS Development Environment

Explanation: You attempted to use a command or qualifier that requires one or more VDE privileges, and you did not have sufficient privileges enabled. This message is normally displayed in conjunction with other error messages, usually a no-privilege error.

User Action: If you are authorized to use the privilege, use the VDE SET PRIVILEGES command to give yourself the necessary privilege. Then try the original operation again. If you are not authorized to use the qualifier, you should either request the necessary privilege from your project administrator or you should not use the qualifier.

NOCREFAC, no CREFAC privilege; cannot create or modify facilities

Facility: VDE, OpenVMS Development Environment

Explanation: You attempted to use a command or qualifier that requires one or more VDE privileges, and you did not have sufficient privileges enabled. This message is normally displayed in conjunction with other error messages, usually a no-privilege error.

User Action: If you are authorized to use the privilege, use the VDE SET PRIVILEGES command to give yourself the necessary privilege. Then try the original operation again. If you are not authorized to use the qualifier, you should either request the necessary privilege from your project administrator or you should not use the qualifier.

NOCREGRP, no CREGRP privilege; cannot create or delete groups

Facility: VDE, OpenVMS Development Environment

Explanation: You attempted to use a command or qualifier that requires one or more VDE privileges, and you did not have sufficient privileges enabled. This message is normally displayed in conjunction with other error messages, usually a no-privilege error.

User Action: If you are authorized to use the privilege, use the VDE SET PRIVILEGES command to give yourself the necessary privilege. Then try the original operation again. If you are not authorized to use the qualifier, you should either request the necessary privilege from your project administrator or you should not use the qualifier.

NOCREMOD, no CREMOD privilege; cannot create or modify modules

Facility: VDE, OpenVMS Development Environment

Explanation: You attempted to use a command or qualifier that requires one or more VDE privileges, and you did not have sufficient privileges enabled. This message is normally displayed in conjunction with other error messages, usually a no-privilege error.

User Action: If you are authorized to use the privilege, use the VDE SET PRIVILEGES command to give yourself the necessary privilege. Then try the original operation again. If you are not authorized to use the qualifier, you should either request the necessary privilege from your project administrator or you should not use the qualifier.

NOCRESCRP, no CRESCRP privilege; cannot create or delete scripts

Facility: VDE, OpenVMS Development Environment

Explanation: You attempted to use a command or qualifier that requires one or more VDE privileges, and you did not have sufficient privileges enabled. This message is normally displayed in conjunction with other error messages, usually a no-privilege error.

User Action: If you are authorized to use the privilege, use the VDE SET PRIVILEGES command to give yourself the necessary privilege. Then try the original operation again. If you are not authorized to use the qualifier, you should either request the necessary privilege from your project administrator or you should not use the qualifier.

NOCRESTEP, no CRESTEP privilege; cannot create or delete steps

Facility: VDE, OpenVMS Development Environment

Explanation: You attempted to use a command or qualifier that requires one or more VDE privileges, and you did not have sufficient privileges enabled. This message is normally displayed in conjunction with other error messages, usually a no-privilege error.

User Action: If you are authorized to use the privilege, use the VDE SET PRIVILEGES command to give yourself the necessary privilege. Then try the original operation again. If you are not authorized to use the qualifier, you should either request the necessary privilege from your project administrator or you should not use the qualifier.

NOCRESTRM, no CRESTRM privilege; cannot create or modify streams

Facility: VDE, OpenVMS Development Environment

Explanation: You attempted to use a command or qualifier that requires one or more VDE privileges, and you did not have sufficient privileges enabled. This message is normally displayed in conjunction with other error messages, usually a no-privilege error.

User Action: If you are authorized to use the privilege, use the VDE SET PRIVILEGES command to give yourself the necessary privilege. Then try the original operation again. If you are not authorized to use the qualifier, you should either request the necessary privilege from your project administrator or you should not use the qualifier.

NOCREUSR, no CREUSER privilege; cannot create or delete users

Facility: VDE, OpenVMS Development Environment

Explanation: You attempted to use a command or qualifier that requires one or more VDE privileges, and you did not have sufficient privileges enabled. This message is normally displayed in conjunction with other error messages, usually a no-privilege error.

User Action: If you are authorized to use the privilege, use the VDE SET PRIVILEGES command to give yourself the necessary privilege. Then try the original operation again. If you are not authorized to use the qualifier, you should either request the necessary privilege from your project administrator or you should not use the qualifier.

NODELARCH, no DELARCH privilege; cannot delete architectures

Facility: VDE, OpenVMS Development Environment

Explanation: You attempted to use a command or qualifier that requires one or more VDE privileges, and you did not have sufficient privileges enabled. This message is normally displayed in conjunction with other error messages, usually a no-privilege error.

User Action: If you are authorized to use the privilege, use the VDE SET PRIVILEGES command to give yourself the necessary privilege. Then try the original operation again. If you are not authorized to use the qualifier, you should either request the necessary privilege from your project administrator or you should not use the qualifier.

NODELDEF, Attempt to delete default prohibited

Facility: VDE, OpenVMS Development Environment

Explanation: You are attempting to delete a object (architecture, stream, facility, etc.) that is a default.

User Action: To make the architecture deleteable use the MODIFY ARCHITECTURE command with the /DELETE qualifier. Then reenter the DELETE ARCHITECTURE command.

NODELFAC, no DELFAC privilege; cannot delete facilities from library

Facility: VDE, OpenVMS Development Environment

Explanation: You attempted to use a command or qualifier that requires one or more VDE privileges, and you did not have sufficient privileges enabled. This message is normally displayed in conjunction with other error messages, usually a no-privilege error.

User Action: If you are authorized to use the privilege, use the VDE SET PRIVILEGES command to give yourself the necessary privilege. Then try the original operation again. If you are not authorized to use the qualifier, you should either request the necessary privilege from your project administrator or you should not use the qualifier.

NODELMOD, no DELMOD privilege; cannot delete modules from library

Facility: VDE, OpenVMS Development Environment

Explanation: You attempted to use a command or qualifier that requires one or more VDE privileges, and you did not have sufficient privileges enabled. This message is normally displayed in conjunction with other error messages, usually a no-privilege error.

User Action: If you are authorized to use the privilege, use the VDE SET PRIVILEGES command to give yourself the necessary privilege. Then try the original operation again. If you are not authorized to use the qualifier, you should either request the necessary privilege from your project administrator or you should not use the qualifier.

NODELREP, no DELREP privilege; cannot delete replacements

Facility: VDE, OpenVMS Development Environment

Explanation: You attempted to use a command or qualifier that requires one or more VDE privileges, and you did not have sufficient privileges enabled. This message is normally displayed in conjunction with other error messages, usually a no-privilege error.

User Action: If you are authorized to use the privilege, use the VDE SET PRIVILEGES command to give yourself the necessary privilege. Then try the original operation again. If you are not authorized to use the qualifier, you should either request the necessary privilege from your project administrator or you should not use the qualifier.

NODELSET, marked /NODELETE

Facility: VDE, OpenVMS Development Environment

Explanation: You are attempting to delete an object (architecture, stream, facility, etc.) that has been marked not-deletable by default or by use of the /NODELETE qualifier.

User Action: To allow the object to be deleted, specify the appropriate MODIFY command with the /DELETE qualifier. Then reenter the failing DELETE command.

NODELSTRM, no DELSTRM privilege; cannot delete streams from library

Facility: VDE, OpenVMS Development Environment

Explanation: You attempted to use a command or qualifier that requires one or more VDE privileges, and you did not have sufficient privileges enabled. This message is normally displayed in conjunction with other error messages, usually a no-privilege error.

User Action: If you are authorized to use the privilege, use the VDE SET PRIVILEGES command to give yourself the necessary privilege. Then try the original operation again. If you are not authorized to use the qualifier, you should either request the necessary privilege from your project administrator or you should not use the qualifier.

NODELTAROOT, no delta-file root directory exists for library

Facility: VDE, OpenVMS Development Environment

Explanation: You have tried to create a directory tree for a specific facility's delta files, but no delta-file root directory exists for the library. Hence VDE cannot create any disk directories for the indicated facility.

User Action: Use the CREATE DIRECTORY_TREE/DELTA_FILES command to create a directory tree for the library's delta files. That will automatically create the delta-file root directory for the library and the delta-file subdirectories for all facilities.

NODIRDELTA, no delta-file directory found for facility *facility*

Facility: VDE, OpenVMS Development Environment

Explanation: VDE tried to access the delta-file directory for the indicated facility, but no such directory is recorded in the database. This normally means that you have not yet created the disk directories for the facility.

User Action: To create the facility's delta-file directory, use the CREATE DIRECTORY_TREE command with the /DELTA_FILES and possibly /FACILITY qualifiers. Use the SHOW LOCATION command to see what disk directories are recorded in the VDE database.

NODIRMARKER, no marker-file directory found for facility *facility*

Facility: VDE, OpenVMS Development Environment

Explanation: VDE tried to access the marker-file directory for the indicated facility, but no such directory is recorded in the database. This normally means that you have failed to create a delta-file subdirectory with directory symbol VDE$MARKER for the indicated facility. Such a directory must exist for you to create modules that use marker files.

User Action: To create the needed subdirectory, use the SET DIRECTORY command with the /DELTA_FILES and /SYMBOL=VDE$MARKER qualifiers. Then create the facility's delta-file directory tree with the CREATE DIRECTORY_TREE command, using the /DELTA_FILES and possibly /FACILITY qualifiers. Use the SHOW LOCATION command to see what disk directories are recorded in the VDE database.

NOFACSPEC, no facility specified for operation

Facility: VDE, OpenVMS Development Environment

Explanation: You have entered a command that requires a default facility to exist, but you have not established a default facility. Usually, this means that the command you have entered contains a module name with an omitted facility name and the facility cannot be defaulted. As a result, the module cannot be looked up in any facility.

User Action: Either establish a default facility with the SET FACILITY command and reenter the command, or reenter the command with an explicitly specified facility name, as in [FACNAM]MOD.TYP.

NOFREE, no free storage available

Facility: VDE, OpenVMS Development Environment

Explanation: VDE has run out of virtual memory.

User Action: Use less complex commands, raise the virtual-address quota, or submit a problem report.

NOFROZEN, no FROZEN privilege; cannot replace modules into frozen streams

Facility: VDE, OpenVMS Development Environment

Explanation: You attempted to use a command or qualifier that requires one or more VDE privileges, and you did not have sufficient privileges enabled. This message is normally displayed in conjunction with other error messages, usually a no-privilege error.

User Action: If you are authorized to use the privilege, use the VDE SET PRIVILEGES command to give yourself the necessary privilege. Then try the original operation again. If you are not authorized to use the qualifier, you should either request the necessary privilege from your project administrator or you should not use the qualifier.

NOGENSTRM, generation *generation* not found in stream *stream*

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated module generation was entered on a command, but no such generation was found in the specified stream.

User Action: This is not necessarily an error; the message simply tells you that the indicated generation does not exist in the specified stream. First, verify that you have either specified the correct stream on the command or that your context is set to the correct stream. If the stream name was correct and you had expected such a generation of a module to exist in that stream, you may use the SHOW GENERATION/STREAM=stream-name command to see what module generations are found in that stream.

NOKEYPAD, unable to set up keypad definitions

Facility: VDE, OpenVMS Development Environment

Explanation: VDE was unable to setup keypad definitions because an error was returned from SMG$DELETE_VIRTUAL_KEYBOAD. VDE uses the SMG facility to perform keypad management.

User Action: Check the setup of the terminal which might cause SMG some problem. If the error still cannot be explained, please contact VDE support for assistance.

NOKEYW, qualifier name is missing - append the name to the slash

Facility: VDE, OpenVMS Development Environment

Explanation: A qualifier is composed of both a "/" and keyword. If only a "/" appears, then this error message is displayed.

User Action: Ensure that there are no dangling "/" in the command.

NOLIBDEF, no VDE library defined

Facility: VDE, OpenVMS Development Environment

Explanation: VDE is unable to perform the specified operation as the target VDE library—the library on which the operation will be performed—has not been established.

User Action: Specify the target VDE library. To determine the available libraries, issue the "SET LIBRARY" command with no parameters. As part of the error message, this command will display the names of the predefined VDE libraries, the Rdb root directory for each, and the mnemonic name for each. Once the mnemnonic name or the Rdb root directory name for the target library is known, one can: A) use the mnemnonic name or the Rdb root directory name as the libname parameter on a "SET LIBRARY libname" VDE command. B) Use use the mnemnonic name or the Rdb root directory name as the parameter on the VDE /LIBRARY=libname. C) Define the logical name VDE$LIBRARY to translate to the VDE mnemnonic name or to the Rdb root directory name.

NOLIBROOM, too many libraries in default file

Facility: VDE, OpenVMS Development Environment

Explanation: VDE has detected too many libraries declared in the VDE.DAT Motif application defaults file, and is unable to display all of the specified libraries in the system library cascading pulldown.

User Action: If you are unable to reduce the number of libraries in the default file, please contact VDE support for assistance.

NOMAIL, no message sent; MAIL messages are disabled

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to indicate that the version of VDE in use contains debug or prototype code, and as a result, no mail message is being sent.

User Action: Contact VDE support for assistance, as the version of VDE in use contains debug or prototype images.

NOMAILSENT, failure to send mail message to user *user*

Facility: VDE, OpenVMS Development Environment

Explanation: VDE has failed in an attempt to send a mail message to the indicated user notifying that user of your action.

User Action: Attempt to resolve the problem based on any other error messages included with this one, and contact VDE support for assistance.

NOMATCHARCH, no architecture name matches the wildcarded string *string*

Facility: VDE, OpenVMS Development Environment

Explanation: You entered a wildcarded architecture name specification on a command, but that specification did not match any architecture names in the VDE database.

User Action: This is not necessarily an error; the message simply tells you that no architectures matching your specification exist in the database. If you had expected such architectures to exist, you can use the SHOW ARCHITECTURE command to see what architectures are present in the database and you can ask your project administator to add any missing architectures.

NOMATCHCTX, no context name matches the wildcarded string *string*

Facility: VDE, OpenVMS Development Environment

Explanation: You entered a wildcarded context name specification on a command, but that specification did not match any context names in the VDE database.

User Action: This is not necessarily an error; the message simply tells you that no contexts matching your specification exist in the database. If you had expected such contexts to exist, you can use the SHOW CONTEXT command to see what contexts are present in the database and you can ask your project administator to add any missing contexts.

NOMATCHFAC, no facility name matches the wildcarded string *string*

Facility: VDE, OpenVMS Development Environment

Explanation: You entered a wildcarded facility name specification on a command, but that specification did not match any facility names in the VDE database.

User Action: This is not necessarily an error; the message simply tells you that no facilities matching your specification exist in the database. If you had expected such facilities to exist, you can use the SHOW FACILITY

command to see what facilities are present in the database and you can ask your project administator to add any missing facilities.

NOMATCHGEN, no generation matches wildcarded string *string*

Facility: VDE, OpenVMS Development Environment

Explanation: You entered a wildcarded generation specification on a command, but that specification did not match any module generations in the VDE database.

User Action: This is not necessarily an error; the message simply tells you that no module generations matching your specification exist in the database. If you had expected such generations to exist, you can use the SHOW GENERATION command to see what generations are present in the database.

NOMATCHGENS, no generation matches wildcarded string *string* in stream *string*

Facility: VDE, OpenVMS Development Environment

Explanation: You entered a wildcarded generation specification on a command, but that specification did not match any module generations in the specified stream.

User Action: This is not necessarily an error; the message simply tells you that no module generations matching your specification exist in the specified stream. First, verify that you have either specified the correct stream on the command or that your context is set to the correct stream. If the stream name was correct and you had expected such generations to exist, you can use the SHOW GENERATION/STREAM=stream-name command to see what generations are present in that stream.

NOMATCHGRP, no group name matches the wildcarded string *string*

Facility: VDE, OpenVMS Development Environment

Explanation: You entered a wildcarded group name specification on a command, but that specification did not match any group names in the VDE database.

User Action: This is not necessarily an error; the message simply tells you that no groups matching your specification exist in the database. If you had expected such groups to exist, you can use the SHOW GROUP command to see what groups are present in the database and you can use the CREATE GROUP command to add any new groups you want.

NOMATCHMOD, no module name matches the wildcarded string *string*

Facility: VDE, OpenVMS Development Environment

Explanation: You entered a wildcarded module name specification on a command, but that specification did not match any module names in the VDE database in the indicated (or in the default) facility. This message

can be generated when double-clicking on an empty facility from within the Motif display.

User Action: This is not necessarily an error; the message simply tells you that no modules matching your specification exist in the VDE database. If you had expected such modules to exist, you can use the SHOW MODULE command to see what modules are present in the database and you can use the CREATE MODULE command to add any missing modules.

NOMATCHREP, no replacement name matches the wildcarded string *string*

Facility: VDE, OpenVMS Development Environment

Explanation: You entered a wildcarded replacement name specification on a command, but that specification did not match any replacement names in the VDE database.

User Action: This is not necessarily an error; the message simply tells you that no replacements matching your specification exist in the database. If you had expected such replacements to exist, you can use the SHOW REPLACEMENT command to see what replacements are present in the database and you can ask your project administator to add any missing replacements.

NOMATCHSCRP, no scripts match the specification

Facility: VDE, OpenVMS Development Environment

Explanation: You requested a display of all scripts of a certains type, possibly matching a wildcarded module-name or facility-name specification, but there are no scripts in the database that match that specification.

User Action: This is not necessarily an error; the message simply tells you that no scripts exist in the database that match your specification.

NOMATCHSES, no session name matches the wildcarded string *string*

Facility: VDE, OpenVMS Development Environment

Explanation: You entered a wildcarded session name specification on a command, but that specification did not match any session names in the VDE database.

User Action: This is not necessarily an error; the message simply tells you that no sessions matching your specification exist in the database. If you had expected such sessions to exist, you can use the SHOW SESSION command to see what sessions are present in the database and you can ask your project administator to add any missing sessions.

NOMATCHSTEP, no build steps for modules matching *string* found in database

Facility: VDE, OpenVMS Development Environment

Explanation: You entered a wildcarded module-name specification to specify build steps, but there are no build steps in the database for modules that match that name specification.

User Action: This is not necessarily an error; the message simply tells you that no build steps exist in the database for modules matching your module-name specification. If you had expected such build steps to exist, you can use the SHOW STEP command to see what build steps are represented in the database.

NOMATCHSTRM, no stream name matches the wildcarded string *string*

Facility: VDE, OpenVMS Development Environment

Explanation: You entered a wildcarded stream name specification on a command, but that specification did not match any stream names in the VDE database.

User Action: This is not necessarily an error; the message simply tells you that no streams matching your specification exist in the database. If you had expected such streams to exist, you can use the SHOW STREAM * command to see what streams are present in the database and you can ask your project administator to add any missing streams.

NOMATCHUSER, no user name matches the wildcarded string *string*

Facility: VDE, OpenVMS Development Environment

Explanation: You entered a wildcarded username specification on a command, but that specification did not match any usernames in the VDE database.

User Action: This is not necessarily an error; the message simply tells you that no users matching your specification exist in the database. If you had expected such users to exist, you can use the SHOW USER * command to see what usernames are present in the database and you can ask your project administator to add any missing users.

NOMEMGRP, no members specified for group *group*

Facility: VDE, OpenVMS Development Environment

Explanation: You have tried to create a group without specifying any members for that group. A group must have at least one member. (The members of a group are the modules to be built when the group name is specified on a GROUP command.)

User Action: Reenter the CREATE GROUP command, specifying the names of the modules that you want to be the members of the new group.

NOMODLIB, no MODLIB privilege; cannot modify attributes of this library

Facility: VDE, OpenVMS Development Environment

Explanation: You attempted to use a command or qualifier that requires one or more VDE privileges, and you did not have sufficient privileges enabled. This message is normally displayed in conjunction with other error messages, usually a no-privilege error.

User Action: If you are authorized to use the privilege, use the VDE SET PRIVILEGES command to give yourself the necessary privilege. Then try the original operation again. If you are not authorized to use the qualifier, you should either request the necessary privilege from your project administrator or you should not use the qualifier.

NOMODMEMGRP, no modules are members of group *group*

Facility: VDE, OpenVMS Development Environment

Explanation: You entered the indicated group name on a command to apply the command to the modules that are members of that group, but the group has no members. Hence the command is not applied to any modules due to that group specification.

User Action: Use the CREATE GROUP or MODIFY GROUP command to specify the members of a group. Use the SHOW GROUP/MEMBER command to determine the membership of a group.

NOMODSTRM, no MODSTRM privilege; cannot modify most stream attributes

Facility: VDE, OpenVMS Development Environment

Explanation: You attempted to use a command or qualifier that requires one or more VDE privileges, and you did not have sufficient privileges enabled. This message is normally displayed in conjunction with other error messages, usually a no-privilege error.

User Action: If you are authorized to use the privilege, use the VDE SET PRIVILEGES command to give yourself the necessary privilege. Then try the original operation again. If you are not authorized to use the qualifier, you should either request the necessary privilege from your project administrator or you should not use the qualifier.

NOMOREMOD, no more modules in review list

Facility: VDE, OpenVMS Development Environment

Explanation: You have attempted to review the next module in the current review list, but there are no more such modules. You also get this message if you attempt to review the previous module when you are positioned at the first module in the list.

User Action: Review build steps in the other direction or exit from the REVIEW REPLACEMENT subsystem.

NOMORESTEP, no more build steps in review list

Facility: VDE, OpenVMS Development Environment

Explanation: You have attempted to review the next build step in the current review list, but there are no more such steps. You also get this message if you attempt to review the previous build step when you are positioned at the first step in the list.

User Action: Review build steps in the other direction or exit from the REVIEW BUILD_JOB subsystem.

NONNULLARG, argument specfied MBZ has non-zero value in routine *VDE-routine*

Facility: VDE, OpenVMS Development Environment

Explanation: VDE has detected a non-fatal internal inconsitency. A non-zero value has been passed as an argument to an internal routine that is specified to have a zero value. This message should only be seen in internal or prototype baselevels of VDE.

User Action: Please notify VDE support. Please include any information that may be useful or relevent in duplicating the problem.

NONOTE, no note topic entered; note entry is disabled

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to indicate that the version of VDE in use contains debug or prototype code, and as a result, that a notes topic was not entered.

User Action: Contact VDE support for assistance, as the version of VDE in use contains debug or prototype images.

NONOTESENT, note NOT posted to conference *conference*

Facility: VDE, OpenVMS Development Environment

Explanation: VDE has failed in an attempt to post a note to the indicated notes conference. Check the conference for general accessability, and - if conference membership is needed, determine if you are a member of the conference from the node on which you are running VDE.

User Action: Attempt to resolve the problem based on any other error messages included with this one, check the documentation for information on setting up the notes conferences, and contact VDE support for assistance.

NOPARSTRM, no parent stream specified for new stream

Facility: VDE, OpenVMS Development Environment

Explanation: You are creating a new development stream, but you did not specify the parent of the new stream. You must specify the parent stream by using the /PARENT qualifier with the CREATE STREAM

command or by establishing a default stream for the VDE session with the SET STREAM command before entering the CREATE STREAM command.

User Action: Establish a default stream with SET STREAM and reenter the CREATE STREAM command or specify the /PARENT qualifier with the CREATE STREAM command.

NOPERFREP, no PREFREP privilege; cannot perform replacements

Facility: VDE, OpenVMS Development Environment

Explanation: You attempted to use a command or qualifier that requires one or more VDE privileges, and you did not have sufficient privileges enabled. This message is normally displayed in conjunction with other error messages, usually a no-privilege error.

User Action: If you are authorized to use the privilege, use the VDE SET PRIVILEGES command to give yourself the necessary privilege. Then try the original operation again. If you are not authorized to use the qualifier, you should either request the necessary privilege from your project administrator or you should not use the qualifier.

NOPNAM, unknown project name specified for the stream

Facility: VDE, OpenVMS Development Environment

Explanation: The specified project name does not match the project names associated with the specified development stream.

User Action: This is a warning message, If you believe that this project should be associated with the stream, please contact the project leader, the system manager(s), or VDE support for assistance.

NOPNAMS, no projects defined for the stream

Facility: VDE, OpenVMS Development Environment

Explanation: There are no projects associated with the specified development stream.

User Action: This is an informational message, If you believe that there should be one or more projects associated with the stream, please contact the project leader, the system manager(s), or VDE support for assistance.

NOPRIVQUAL, insufficient privilege to use *qualifer* qualifier

Facility: VDE, OpenVMS Development Environment

Explanation: You attempted to use a command qualifier that requires a privilege, and you did not have that privilege set.

User Action: If you are authorized to use the qualifier, use the SET PRIVILEGES command to give yourself the necessary privilege. Then try the original operation again. If you are not authorized to use the qualifier, you should either request the necessary privilege from your project administrator or you should not use the qualifier.

NOPRIVS, insufficient VDE privilege(s) for attempted operation

Facility: VDE, OpenVMS Development Environment

Explanation: You attempted to use a command or qualifier that requires one or more VDE privileges, and you did not have sufficient privileges enabled. This message is normally displayed in conjunction with other error messages, usually a no-privilege error.

User Action: If you are authorized to use the privilege, use the VDE SET PRIVILEGES command to give yourself the necessary privilege. Then try the original operation again. If you are not authorized to use the qualifier, you should either request the necessary privilege from your project administrator or you should not use the qualifier.

NOPROMPT, no prompt specified; using default prompt

Facility: VDE, OpenVMS Development Environment

Explanation: This is an informational message, and indicates that no prompt was specified on the SET PROMPT command. As no prompt was specified, VDE will revert to the default VDE prompt.

User Action: No action is required.

NORESFORMOD, no reservations exist for module *module*

Facility: VDE, OpenVMS Development Environment

Explanation: You have tried to reference reservations for the indicated module or modules (the name may be wildcarded), but there are no reservations in the VDE database for that module or those modules.

User Action: This is not necessarily an error; the message simply tells you that the indicated module or modules are not reserved.

NORESREP, no RESREP privilege; cannot reserve and replace modules

Facility: VDE, OpenVMS Development Environment

Explanation: You attempted to use a command or qualifier that requires one or more VDE privileges, and you did not have sufficient privileges enabled. This message is normally displayed in conjunction with other error messages, usually a no-privilege error.

User Action: If you are authorized to use the privilege, use the VDE SET PRIVILEGES command to give yourself the necessary privilege. Then try the original operation again. If you are not authorized to use the qualifier, you should either request the necessary privilege from your project administrator or you should not use the qualifier.

NOSCRATCH, no SYS$SCRATCH translation

Facility: VDE, OpenVMS Development Environment

Explanation: The translation of the SYS$SCRATCH logical name is not valid.

User Action: Reenter the VDE command after redefining SYS$SCRATCH to a valid directory specification.

NOSCRIPTYP, no script type specified on command

Facility: VDE, OpenVMS Development Environment

Explanation: The current SCRIPT command (such as CREATE SCRIPT) does not include any qualifier to specify the type of script that the command should apply to. As a result, the command cannot be performed.

User Action: Reenter the command with a qualifier to specify the script type of the script.

NOSCT, replacement cause not known; module not replaced

Facility: VDE, OpenVMS Development Environment

Explanation: VDE was not provided with the cause of the replacement being performed. The cause of a replacement—usually a Source Code Tracking (SCT) entry or a source code Fold—is tracked.

User Action: Specify the SCT note via the /SCT or /NOTE qualifiers on the REPLACE command, or by providing answers to teh replacement prompts within VDE.

NOSCTSRC, source file *record* for SCT note not found

Facility: VDE, OpenVMS Development Environment

Explanation: VDE is not able to locate the source file for the SCT note.

User Action: Respecify the command, using a file that exists.

NOSETLIB, the SET LIBRARY command is locked out

Facility: VDE, OpenVMS Development Environment

Explanation: The SET LIBRARY command is disabled when executing a beta version of VDE.

User Action: Contact VDE support for assistance, as the version of VDE in use contains beta-only images.

NOSETTERM, the SET TERMINAL command is not supported on this terminal

Facility: VDE, OpenVMS Development Environment

Explanation: SET TERMINAL is only supported on VT52, VT100-series, VT200-series, VT300-series, VT400-series, VT500-series, and DECterm terminals.

NOSPAWN, spawn command failed

Facility: VDE, OpenVMS Development Environment

Explanation: A process cannot be spawned by VDE.

User Action: Check your process's quotas.

NOSTAGROOT, no staging area root directory exists for library

Facility: VDE, OpenVMS Development Environment

Explanation: You have tried to create queued replacement for some module, but no staging area root directory exists for the library. Hence VDE cannot create a disk directory for saving (or ''staging'') the files to be replaced.

User Action: Use the CREATE DIRECTORY_TREE/STAGING command to create the root directory for the library's staging areas. Then reenter the REPLACE command. Alternatively, replace the module with an immediate (as opposed to queued) replacement. An immediate replacement does not use a staging area.

NOSTRMROOT, no stream root directory exists for stream *stream*

Facility: VDE, OpenVMS Development Environment

Explanation: You have tried to create a directory tree for a specific facility within the current stream, but no root directory exists for the stream. Hence VDE cannot create any disk directories for the indicated facility.

User Action: Use the CREATE DIRECTORY_TREE command to create a directory tree for the current stream. That will automatically create both the stream root directory and the subdirectories for all facilities.

NOSTRMSPEC, no stream specified for operation

Facility: VDE, OpenVMS Development Environment

Explanation: The command you have entered requires that you specify the name of a development stream, either with the /STREAM qualifier or as a command parameter, but the stream name is missing on the entered command.

User Action: Renter the command, specifying the stream name as required.

NOSUCHARCH, architecture *architecture* not found in database

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated architecture name was entered on a command, but no such architecture was found in the VDE database.

User Action: This is not necessarily an error; the message simply tells you that the indicated architecture does not exist in the database. However, if you had expected such an architecture to exist, you may use the SHOW ARCHITECTURE command to see what architectures are found in the database. If the desired architecture is not there, ask your project administator to add it to the database. Then reenter your command with a valid architecture name.

NOSUCHCTX, context *context* not found in database

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated context name was entered on a command, but no such context was found in the VDE database.

User Action: This is not necessarily an error; the message simply tells you that the indicated context does not exist in the database. However, if you had expected such a context to exist, you may use the SHOW CONTEXT command to see what contexts are found in the database.

NOSUCHDIRSYM, directory symbol *symbol* not found in database

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated directory symbol was referenced by a script but no such directory symbol was found in the VDE database. VDE tried to use the directory symbol either to look up the directory location for the script's DCL command file or to evaluate a script function such as $DIR or $STRMDIR. The most likely reason for the error is that you used a script function that takes a directory symbol as a parameter, and you misspelled the symbol name or specified a symbol you have not yet defined. Directory symbols are defined with the /SYMBOL qualifier of the SET DIRECTORY command.

User Action: Correct the script function call to have a correct directory symbol that is already defined. Then reenter the script in the database with the CREATE SCRIPT command and reenter the command that got the error message.

NOSUCHFAC, facility *facility* not found in database

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated facility name was entered on a command, but no such facility was found in the VDE database.

User Action: This is not necessarily an error; the message simply tells you that the indicated facility does not exist in the database. However, if you had expected such a facility to exist, you may use the SHOW FACILITY command to see what facilities are found in the database. If the desired facility is not there, ask your project administator to add it to the database. Then reenter your command with a valid facility name.

NOSUCHFOLD, fold record *record* not found for module *module* and stream *stream*

Facility: VDE, OpenVMS Development Environment

Explanation: You have tried to reference a fold record with the indicated fold identifier for the indicated module and stream, but there is no such fold record in the VDE database. As a result, VDE has aborted the current command and rolled back the current transaction.

User Action: Specify a valid fold record and reenter the command.

NOSUCHFUNC, no such script function exists: *function*

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated script function was specified in a VDE script, but no such function exists. The most likely cause for this error is that you misspelled the function name. VDE recognizes a script function by the fact that all script function names begin with a dollar sign. Directory symbol names, which can appear in the same places as functions in scripts, should not begin with dollar signs.

User Action: Correct the spelling of the function name or select a directory symbol name that does not begin with a dollar sign.

NOSUCHGEN, generation *generation* not found in database

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated module generation was entered on a command, but no such generation was found in the VDE database.

User Action: This is not necessarily an error; the message simply tells you that the indicated generation does not exist in the database. However, if you had expected such a generation of a module to exist, you may use the SHOW GENERATION command to see what module generations are found in the database.

NOSUCHGRP, group *group* not found in database

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated group name was entered on a command, but no such group was found in the VDE database.

User Action: This is not necessarily an error; the message simply tells you that the indicated group does not exist in the database. However, if you had expected such a group to exist, you may use the SHOW GROUP command to see what groups are found in the database. If the desired group is not there, use the CREATE GROUP command to add it to the database. Then reenter your command with a valid group name.

NOSUCHMOD, module *module* not found in database

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated module name was entered on a command, but no such module was found in the VDE database.

User Action: This is not necessarily an error; the message simply tells you that the indicated module does not exist in the database. However, if you had expected such a module to exist, you may use the SHOW MODULE command to see what modules are found in the database. If the desired module is not there, you can use the CREATE MODULE command to add it to the database. Then reenter your command with a valid module name.

NOSUCHPRIV, no such privilege name exists: *priv-name*

Facility: VDE, OpenVMS Development Environment

Explanation: An invalid privilege name was specified on a VDE command.

User Action: Check the spelling of the privilege name and reenter the command with a correct privilege name.

NOSUCHQUE, batch queue *queue* does not exist; check spelling

Facility: VDE, OpenVMS Development Environment

Explanation: You have specified a batch queue on a BUILD command or a related command, but that queue does not exist on the system. Possibly you have misspelled the queue name. Hence build jobs cannot be submitted to that batch queue.

User Action: Specify a valid queue name and restart the build job.

NOSUCHREP, replacement *replacement* not found in database

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated replacement name was entered on a command, but no such replacement was found in the VDE database.

User Action: This is not necessarily an error; the message simply tells you that the indicated replacemen does not exist in the database. However, if you had expected such a replacemen to exist, you may use the SHOW REPLACMENT command to see what replacements are found in the database. If the desired replacement is not there, ask your project administator to add it to the database. Then reenter your command with a valid replacement name.

NOSUCHRES, reservation *reservation* for module *module* not found

Facility: VDE, OpenVMS Development Environment

Explanation: You have specified a reservation identifier for a module reservation, but no reservation with that identifier exists. Hence the current command has not been performed for that reservation.

User Action: Specify the correct reservation identifier and reissue the command.

NOSUCHSCRP, script *script* not found in database

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated script was specified on a command, but no such script is found in the VDE database.

User Action: This is not necessarily an error; the message simply tells you that the indicated script does not exist in the database. However, if you had expected such a script to exist, you may use the SHOW SCRIPT command to see what scripts are present in the database.

NOSUCHSES, session *session* not found in database

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated session name was entered on a command, but no such session was found in the VDE database.

User Action: This is not necessarily an error; the message simply tells you that the indicated session does not exist in the database. However, if you had expected such a session to exist, you may use the SHOW SESSION command to see what session are found in the database. If the desired session is not there, ask your project administator to add it to the database. Then reenter your command with a valid session name.

NOSUCHSTEP, no build step for module *module* found in database

Facility: VDE, OpenVMS Development Environment

Explanation: No compile or link build step is recorded in the database for the indicated module.

User Action: This is not necessarily an error; the message simply tells you that the indicated build step is not represented in the database. However, if you had expected such a build step to exist, you may use the SHOW STEP command to see what build steps are found in the database.

NOSUCHSTRM, stream *stream* not found in database

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated stream name was entered on a command, but no such stream was found in the VDE database.

User Action: This is not necessarily an error; the message simply tells you that the indicated stream does not exist in the database. However, if you had expected such a stream to exist, you may use the SHOW STREAM * command to see what streams are found in the database. If the desired stream is not there, ask your project administator to add it to the database. Then reenter your command with a valid stream name.

NOSUCHSYM, no such symbol '*text*' exists

Facility: VDE, OpenVMS Development Environment

Explanation: No symbols matching the given symbol specification have been found.

User Action: Check the spelling of the symbol specification in question. The SHOW SYMBOL * command can also be used to display all the symbol names.

NOSUCHUSER, user *user* not found in database

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated username was entered on a command, but no such user was found in the VDE database.

User Action: This is not necessarily an error; the message simply tells you that the indicated user does not exist in the database. However, if you had expected such a user to exist, you may use the SHOW USER *

command to see what usernames are found in the database. If the desired user is not there, ask your project administator to add that user to the database. Then reenter your command with a valid username.

NOTCONNLIB, you are not connected to a VDE library; command not performed

Facility: VDE, OpenVMS Development Environment

Explanation: You are trying to perform a command that requires you to be connected to an existing VDE library, but you are not connected to such a library. This normally means that you have not defined logical name VDE$LIBRARY to point to the root directory of a properly formed VDE library that you are authorized to use. This problem can also occur after you have encountered an error when trying to set or create a new library; you are then disconnected from the formerly set library without successfully connecting to the new one, leaving you unconnected to any library.

User Action: Define logical name VDE$LIBRARY to point to the root directory for the VDE library you wish to access. Then try the command again.

NOTEID, topic *major.minor* entered in conference

Facility: VDE, OpenVMS Development Environment

Explanation: This message indicates that VDE has entered a note into a notes conference on behalf of the user.

User Action: No action is required.

NOTESCONF, accessing notes conference *conference*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that VDE has successfully opened the indicated notes conference for notes access.

User Action: None

NOTESENT, note posted in conference *conference*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that VDE has successfully posted a note in the indicated notes conference. This message is issued when you use the /LOG qualifier on a command that sends notifications to notes conferences.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

NOTESENTID, note *noteID* posted in conference *conference*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that VDE has successfully posted a note in the indicated notes conference. This message is issued when you use the /LOG qualifier on a command that sends notifications to notes conferences.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

NOTESERR, error from DEC Notes at *routine-name*

Facility: VDE, OpenVMS Development Environment

Explanation: VDE has encountered an error while using the notes callable interface.

User Action: Attempt to resolve the problem based on any other error messages included with this one, check the documentation for information on setting up the notes conferences, and contact VDE support for assistance.

NOTESID, unable to parse DEC Notes topic *note-number*

Facility: VDE, OpenVMS Development Environment

Explanation: VDE has encountered an error while using the notes callable interface.

User Action: Attempt to resolve the problem based on any other error messages included with this one, check the documentation for information on setting up the notes conferences, and contact VDE support for assistance.

NOTEXTSTR, text string argument missing in call to *routine-name*

Facility: VDE, OpenVMS Development Environment

Explanation: VDE has detected an inconsistency within an internal routine which uses the notes callable interface.

User Action: Please contact VDE support for assistance. Please include any information that may be useful or relevent in duplicating the problem.

NOTIMPL, this feature is not currently implemented

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated feature or callback is not currently implemented.

User Action: Please contact VDE support for assistance.

NOTINRANGE, integer value *number* not in valid range of *low-bound* to *high-bound*

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated integer value is not within the valid range for the qualifier or parameter on which it is specified.

User Action: Reenter the command with the integer value within the proper range.

NOTPROP, generation *generation* not propagated past stream *stream*

Facility: VDE, OpenVMS Development Environment

Explanation: Because the /PROPAGATE or /NOPROPAGATE qualifier was specified on the REPLACE command, the indicated module generation is not propagated past the indicated development stream.

User Action: None. This is an informational message that merely confirms that VDE performed change propagation as requested.

NOTRESINSTRM, module *module* not reserved in stream *stream*

Facility: VDE, OpenVMS Development Environment

Explanation: You are trying to unreserve or replace a module in the indicated stream, but the module is not reserved in that stream.

User Action: Use the SHOW RESERVATION/FULL command to see what stream the reservation was made in. Then use the SET STREAM command to set the current stream to the appropriate stream and re-enter the command, or (when available) you can append the /STREAM qualifier on the failing command and reissue the command. If you are performing a REPLACEMENT, a replacement performed in the ancestor stream propogated forward into any and successors of the target stream.

NOTREVREP, you are not a reviewer for replacement *replacement*

Facility: VDE, OpenVMS Development Environment

Explanation: You are trying to mark a replacement as accepted or rejected, but you are not a reviewer for the replacement. As a result, your attempt to mark the replacement as accepted or rejected has failed.

User Action: Specify the name of a replacement for which you are a reviewer or have yourself added as a reviewer by someone authorized to add you.

NOUNCLOSE, no UNCLOSE privilege; cannot unclose a closed stream

Facility: VDE, OpenVMS Development Environment

Explanation: You attempted to use a command or qualifier that requires one or more VDE privileges, and you did not have sufficient privileges enabled. This message is normally displayed in conjunction with other error messages, usually a no-privilege error.

User Action: If you are authorized to use the privilege, use the VDE SET PRIVILEGES command to give yourself the necessary privilege. Then try the original operation again. If you are not authorized to use

the qualifier, you should either request the necessary privilege from your project administrator or you should not use the qualifier.

NOUSERNAME, no USERNAME privilege; cannot reserve and replace for other users

Facility: VDE, OpenVMS Development Environment

Explanation: You attempted to use a command or qualifier that requires one or more VDE privileges, and you did not have sufficient privileges enabled. This message is normally displayed in conjunction with other error messages, usually a no-privilege error.

User Action: If you are authorized to use the privilege, use the VDE SET PRIVILEGES command to give yourself the necessary privilege. Then try the original operation again. If you are not authorized to use the qualifier, you should either request the necessary privilege from your project administrator or you should not use the qualifier.

NOVDEDEVO, no VDEDEVO privilege; access to dungeon restricted

Facility: VDE, OpenVMS Development Environment

Explanation: You attempted to use a command or qualifier that requires one or more VDE privileges, and you did not have sufficient privileges enabled. This message is normally displayed in conjunction with other error messages, usually a no-privilege error.

User Action: If you are authorized to use the privilege, use the VDE SET PRIVILEGES command to give yourself the necessary privilege. Then try the original operation again. If you are not authorized to use the qualifier, you should either request the necessary privilege from your project administrator or you should not use the qualifier.

NULSTRNOT, a null string is not allowed

Facility: VDE, OpenVMS Development Environment

Explanation: This qualifier requires that a string with a minimum length of 1 character be provided if any string is specified at all. The null string is not allowed.

User Action: Reenter the command providing a string with this qualifier, or no value at all.

NYI, feature not yet implemented

Facility: VDE, OpenVMS Development Environment

Explanation: VDE has detected a request for a feature that has not yet been implemented.

User Action: Please contact VDE support for assistance. Please include any information that may be useful or relevent in duplicating the problem.

OFFLINE, source code control system offline, try again later

Facility: VDE, OpenVMS Development Environment

Explanation: The VDE system is offline for maintenance or upgrade, please try your request again later.

User Action: The VDE system is offline fopr maintenance or upgrade, please try again later. If you have questions, please contact VDE support or the system management staff for assistance.

OUTFILE, output file is *file*

Facility: VDE, OpenVMS Development Environment

Explanation: Indicates the output file of the current command. The FETCH and RESERVE commands display this message to show the location of the output file containing the fetched module generation.

User Action: None. This is an informational message.

OUTMODSRC, output module *module* of build step is a source module; ignored

Facility: VDE, OpenVMS Development Environment

Explanation: The automatically collected dependency information for the current build step specifies that the indicated module is an output from the step. However, the module is marked as a source module in the VDE database, and a source module cannot be an output of a build step. Hence this dependency information is ignored and is not recorded in the database.

User Action: If the module is erroneously marked as a source module in the VDE database, you should delete the module and rerun the build step. If the module actually is a source module, you should investigate why the compiler or other processor specifies it as an output from the build step.

OUTNLA0, if /OUTPUT=NLA0: was used, respecify with non-null output

Facility: VDE, OpenVMS Development Environment

Explanation: This message suggests a workaround for a failure to fetch a module—this situation occurs only when fetching or reserving a module that uses a marker file, and when /OUTPUT=NLA0: was specified on the command. The underlying problem will be fixed in a future release of VDE.

User Action: If fetching or reserving a module that uses a marker file, respecify the command with a non-null output device and directory.

OVERRIDE, VDE library controls overridden via OpenVMS privileges

Facility: VDE, OpenVMS Development Environment

Explanation: An attempt has been made to perform an operation that VDE would normally prohibit, but VDE has allowed the operation because of OpenVMS privileges possessed by the caller.

User Action: No action is required.

OVFLOWBUF, overflow of the input buffer, command ignored.

Facility: VDE, OpenVMS Development Environment

Explanation: A VDE command line may only consist of records that are less than 1024 characters long.

User Action: If you want more than 1024 characters for input, divide each line into records that are less than 1024 characters and use the continuation character "-" at the point of each break. If the sum of these records are greater than 32,768 characters, then the complete command line cannot be processed.

PREFIXNOTALL, prefix VDE$ not allowed in facility or stream names

Facility: VDE, OpenVMS Development Environment

Explanation: You have tried to create or rename a facility or stream with a name that begins with the prefix ‘‘VDE$’’ but VDE does not allow facilities or streams to have such names. VDE uses facility and stream names as disk directory names, but VDE also creates directories reserved for its own use whose names always begin with the VDE$ prefix. To prevent name conflicts between facility or stream directories and these reserved directories, VDE does not allow facility or stream names to begin with the VDE$ prefix.

User Action: Reenter the command with a facility or stream name that does not begin with VDE$.

PROTOLIB, using prototype library *library*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to indicate that the version of VDE in use contains debug or prototype code, and as a result, that a particular library is being used.

User Action: Contact VDE support for assistance, as the version of VDE in use contains debug or prototype images.

PROTOTYPE, this is a prototype version of VDE

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to indicate that the version of VDE in use contains debug or prototype code.

User Action: Contact VDE support for assistance, as the version of VDE in use contains debug or prototype images.

QARDBSYNTAX, invalid QAR database name syntax: *name*

Facility: VDE, OpenVMS Development Environment

Explanation: The name or format specified for the QAR database name is not using valid syntax.

User Action: Respecify the QAR database name using only alphanumeric, underscore, dollar sign and hyphen characters.

QCHECKCOUNT, queued.checkin.count: *count*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

QDATETIME, queued.checkin.datetime: *datetime*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

QERRORS, queued.checkin.defectsfixed: *defects*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

QGROUP, queued.checkin.group: *group*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

QITEM, queued.checkin.item: *item*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

QLIBMNE, queued.library.mnemonic: *libmnemonic*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

QMODULE, queued.checkin.module: *modname*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

QNOTE, queued.checkin.note: *topic.reply*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

QPROJECT, queued.project.name: *project*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

QREASON, queued.checkin.reasonmask: *reason*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

QREPLID, queued.checkin.replacementid: *replid*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

QREPLNAME, queued.checkin.replname: *replname*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

QRMPBCOUNT, queued.module.count: *count*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

QRMPBMODFLAG, queued.module.modflags: *modflags*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

QRMPBMODNAME, queued.module.modname: *modname*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

QRMPBREPLREMARK, queued.module.replaceremark: *replaceremark*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

QRMPBRESREMARK, queued.module.reserveremark: *reserveremark*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

QRMPBSTAGING, queued.module.staging: *staging*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

QSTREAM, queued.checkin.stream: *stream*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

QTYPCOD, queued.checkin.typecode: *type*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

QTYPNAM, queued.checkin.typename: *type*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

QUALDISALL, qualifier *qualifier* is disallowed for the *command* command

Facility: VDE, OpenVMS Development Environment

Explanation: You have specified a command qualifier that is not allowed with the specified command. Because the qualifier is not allowed, the command is not executed. In certain cases, the qualifier is disallowed by your VDE library due to a library attribute, and the qualifier would be allowed if the library attribute that disallows it were changed.

User Action: Reenter the command without the offending qualifier or change the library attribute that disallows it, if applicable.

QUALNOTALL, qualifier *qualifier* is not allowed when a parameter is specified

Facility: VDE, OpenVMS Development Environment

Explanation: You have specified both the indicated qualifier and a command parameter on a command that does not allow both to be specified. The command is not executed for this reason.

User Action: Reenter the command with either the indicated qualifier or with command parameters, but not both.

QUALTOOMANY, qualifier *qualifier* has been specified too many times

Facility: VDE, OpenVMS Development Environment

Explanation: You have specified the indicated qualifier too many times for a parameter on this command line. The command is not executed for this reason.

User Action: Reenter the command with the indicated qualifier specified no more than once for each parameter.

QUECLOSED, batch queue *queue* is closed and cannot accept jobs

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated batch queue is closed and cannot accept batch jobs. It thus cannot be used to run the build job you are starting.

User Action: You can either run the build job on the remaining batch queues you have specified, or you can restart the build job and specify a different set of batch queues for it.

QUEPAUSED, batch queue *queue* is paused or pausing; job execution is halted

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated batch queue is paused or in the process of pausing. You can still submit batch jobs to the queue, but the jobs will not execute until the pause condition ends.

User Action: You can either let the build job be queued to the paused batch queue and wait for the pause condition to end, or you can restart the build job and specify a different set of batch queues for it.

QUESTOPPED, batch queue *queue* is stopped or stopping; job execution is halted

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated batch queue is stopped or in the process of stopping. You can still submit batch jobs to the queue, but the jobs will not execute until the queue is started by a system manager or operator.

User Action: You can either let the build job be queued to the stopped batch queue and wait for the queue to be started, or you can restart the build job and specify a different set of batch queues for it.

QUOCHECK, performing a quota check operation

Facility: VDE, OpenVMS Development Environment

Explanation: This message indicates that the VDE/READ_UAF image is checking process quotas in the SYSUAF, and that it will NOT update quota settings in the SYSUAF.

User Action: Informational, no action is necessary

QUOLOW, quota *quota* set to: *old*, recommended: *new*

Facility: VDE, OpenVMS Development Environment

Explanation: This message indicates that the VDE/READ_UAF image has detected a username with a quota value that is below the recommended settings.

User Action: Contact your local system manager, and ask that the quotas on the specified user be checked against the VDE minimum values, and raised where necessary. The system manager or VDE database administrator can use the VDE command VDE/READ_UAF/WRITE_UAF to raise the process quotas to the required minimum values on all VDE users. (Process quota values above the required minimums will not be altered.)

QUORESET, performing a quota check and update operation

Facility: VDE, OpenVMS Development Environment

Explanation: This message indicates that the VDE/READ_UAF image is checking process quotas in the SYSUAF, and that it WILL update quota settings in the SYSUAF.

User Action: Informational, no action is necessary

QUOSTREXP, quoted string expected

Facility: VDE, OpenVMS Development Environment

Explanation: A quoted string was expected. This quoted string must be delimited by either single or double quotes.

QUOUSER, user *user* below recommended quota setting(s)

Facility: VDE, OpenVMS Development Environment

Explanation: This message indicates that the VDE/READ_UAF image has detected a username with a quota value that is below the recommended settings.

User Action: Contact your local system manager, and ask that the quotas on the specified user be checked against the VDE minimum values, and raised where necessary. The system manager or VDE database administrator can use the VDE command VDE/READ_UAF/WRITE_UAF to raise the process quotas to the required minimum values on all VDE users. (Process quota values above the required minimums will not be altered.)

QUSERNAME, queued.checkin.username: *username*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message. ! ==========
==========

READERR, VDE input read error, forced to exit

Facility: VDE, OpenVMS Development Environment

Explanation: An error was found in reading an input line.

User Action: All command line information is read from the logical VDE$INPUT. Something is interfering with the reading from the file or device that is specified by this logical name. Determine where VDE$INPUT is pointing. If you cannot determine what is causing the interference, please submit a problem report.

REGRESSION, operating in mode suitable for regression tests

Facility: VDE, OpenVMS Development Environment

Explanation: This message indicates that the VDE image has detected a request to switch into a mode suitable for operations within a VDE regression test suite. This request can be initiated only by a user with OpenVMS privileges, or by a VDE image linked with this mode enabled.

User Action: Contact VDE support for assistance, as this message should not be displayed under normal operations.

REPACCEPT, replacement *replacement* marked as accepted in database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the VDE database now records that you have accepted the indicated replacement. This message is only issued when you use the /LOG qualifier on the ACCEPT REPLACEMENT command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

REPADDED, replacement *replacement* added to the database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated replacement definition has been added to the VDE database. This message is only issued when you use the /LOG qualifier on the REPLACE command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

REPBELUSER, replacement *replacement* belongs to another user

Facility: VDE, OpenVMS Development Environment

Explanation: You are attempting to modify or cancel a queued replacement that belongs to another user, and you do not have the PERFREP privilege that would allow you to cancel or modify another user's replacements. As a result, the replacement is not cancelled or modified.

User Action: Reenter the command specifying your own queued replacement to cancel or modify, or else have someone with the necessary privilege perform the operation.

REPCAN, replacement *replacement* cancelled

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated replacement has been cancelled. This message is only issued when you use the /LOG qualifier on the CANCEL REPLACEMENT command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

REPDEL, replacement *replacement* deleted from the database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the definition for the indicated replacement has been deleted from the VDE database. This message is only issued when you use the /LOG qualifier on the DELETE REPLACEMENT command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

REPEXISTS, replacement *replacement* already exists; replacement not changed

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated replacement name was specified on a REPLACE command, but an entry for that replacement is already present in the database. Hence a new entry was not created for that replacement. Alternatively, the indicated name was specified as the new name for an existing replacement on the MODIFY REPLACEMENT command, but another replacement already has the requested new name. Hence the name of the existing replacement could not be changed to the new name.

User Action: If you entered the wrong replacement name, reenter the command with the correct name. If you entered the right replacement name, use the SHOW REPLACEMENT command to display the database information for that replacement. If you are not satisfied with that information, use the MODIFY REPLACEMENT command to change it.

REPLACED, generation *generation* replaced into stream *stream*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated module has been replaced into the indicated development stream in the current VDE library. It has also been assigned the indicated generation number. This message is only issued when you use the /LOG qualifier with the REPLACE command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

REPLQUE, module *module* queued for replacement *replacement*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated module has been queued for replacement as a member of the named replacement. This message is only issued when you use the /LOG qualifier on the REPLACE command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

REPMOD, information for replacement *replacement* modified in the database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that information for the indicated replacement has been modified in the VDE database. This message is only issued when you use the /LOG qualifier on the MODIFY REPLACEMENT command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

REPNOTFOUND, replacement *replacement* not found in database; parameter ignored

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated repalcement name was entered on a SHOW REPLACEMENT command, but no such replacement was found in the VDE database.

User Action: This is not necessarily an error; the message simply tells you that the indicated replacement does not exist in the database.

REPNOTPERF, replacement *replacement* for stream *stream* not performed; does not match stream *stream*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated replacement has not been performed because it belongs to a stream that does not match the stream name you specified with the /STREAM qualifier to the PERFORM REPLACEMENT command (or your default stream). This message is only issued when you use the /LOG qualifier on the PERFORM REPLACEMENT command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

REPPERF, replacement *replacement* performed

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated replacement has been performed. This message is only issued when you use the /LOG qualifier on the PERFORM REPLACEMENT command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

REPREJECT, replacement *replacement* marked as rejected in database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the VDE database now records that you have rejected the indicated replacement. This message is only issued when you use the /LOG qualifier on the REJECT REPLACEMENT command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

REPSYNTAX, invalid replacement name syntax: *name*

Facility: VDE, OpenVMS Development Environment

Explanation: The name or format specified for the replacement name is not using valid syntax.

User Action: Respecify the replacement name using only alphanumeric, underscore, dollar sign and hyphen characters.

REQQUAMIS, required command qualifier missing

Facility: VDE, OpenVMS Development Environment

Explanation: Both the DEFINE and DELETE commands must have a qualifier to specify the type of definition or deletion. You must use either /KEY or /COMMAND.

RESDEL, reservation *reservation* of *module* for user *user* has been deleted

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated reservation has been deleted from the VDE database because you performed an operation that changed what generation is the latest generation of the module for some stream. Such operations include the DELETE, INSERT, and REMOVE GENERATION commands. After such an operation, the reservation is based on the wrong generation and can no longer be replaced into the stream.

User Action: None. The affected users will be informed that their reservations are deleted. They should reserve the module again and redo their changes.

RESERVED, generation *generation* reserved from stream *stream*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated module and generation has been reserved from the current library. This message is only issued when you use the /LOG qualifier on the RESERVE command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

RESERVSES, generation *generation* reserved from stream *stream* for session *session*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated module and generation has been reserved from the VDE library and is part of the indicated reservation session. This message is only issued when you use the /LOG qualifier on the RESERVE command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

RESIDREQ, reservation identifier required for module *module*

Facility: VDE, OpenVMS Development Environment

Explanation: You have multiple reservations for the indicated module. You must therefore specify a reservation identifier with the /IDENTIFICATION qualifier to uniquely indicate to which reservation you want the command to apply.

User Action: Reissue the command with the /IDENTIFICATION qualifier to indicate the specific reservation to which you want the command to apply.

RESIDUSED, reservation Id *ID* already used for module *module* attempt to reserve module has failed

Facility: VDE, OpenVMS Development Environment

Explanation: You are trying to reserve a module concurrently using the same reservation identification as you have used on a previous reservation of the same module in the same stream development stream. This is not allowed, since reservation identifications must be unique.

User Action: Reissue the RESERVE command with a different reservation identification specified on the /IDENTIFICATION qualifier. Alternatively, you can omit the /IDENTIFICATION qualifier, in which case VDE will assign a unique identification for you.

RESMOD, reservation *reservation* for module *module* modified in database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated reservation record for the indicated module has been modified in the VDE database. This message is only issued when you use the /LOG qualifier on the MODIFY RESERVATION command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

RESQUEUED, reservation request for *module* queued for stream *stream*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that VDE has queued a reservation request for the indicated module and development stream. This message is only issued when you use the /LOG qualifier on the RESERVE command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

RETURNED, control returned to process *process-name*

Facility: VDE, OpenVMS Development Environment

Explanation: After performing either a SPAWN or ATTACH operation, this message will be generated to inform you when you have returned to VDE's process.

RIGIDNOTFOU, required rights identifier *ID* not found in rights database

Facility: VDE, OpenVMS Development Environment

Explanation: You have not yet been added to the VDE database, and to be added you must hold the indicated VMS rights identifier. However, that rights identifier is not present in the system rights database. As a result, you are not allowed access to the VDE library.

User Action: Have your system manager add the rights identifier to the system rights database and assign it to the users who should have access to the VDE library, or have the VDE library maintainer use the MODIFY LIBRARY command specify the correct rights identifier in the VDE database.

RIGIDNOTHELD, you do not hold required rights identifier *ID*

Facility: VDE, OpenVMS Development Environment

Explanation: You have not yet been added to the VDE database, and to be added you must hold the indicated VMS rights identifier. However, you do not hold that rights identifier in the system rights database. As a result, you are not allowed access to the VDE library.

User Action: If you should have access to the VDE library, have your system manager assign you the indicated rights identifier. Then try accessing the VDE library again.

ROLLBACK, database transaction has been rolled back

Facility: VDE, OpenVMS Development Environment

Explanation: The database transaction associated with the current command has been rolled back because an error occurred while processing the command. This means that all changes the command would have made to the database if successful have in fact been removed from the database.

User Action: Correct the error and retry the command.

SCRDONE, completed wait for script

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that VDE has completed a wait for a script.

User Action: Contact VDE support for assistance, as the version of VDE in use contains debugging images.

SCRIPTONLY, VDE/STATUS is invoked only within scripts

Facility: VDE, OpenVMS Development Environment

Explanation: This message is indicates the user has attempted to invoke the VDE/STATUS command at the DCL prompt; this command is for script processing within VDE and is not intended to directly accessed by users.

User Action: Don't do this.

SCRLOAD, loading script into subprocess

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that VDE has loaded a script into a subprocess for execution.

User Action: Contact VDE support for assistance, as the version of VDE in use contains debugging images.

SCRPADDED, script from file *file* added to the database script *script*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated script has been added to the VDE database from the indicated file. The script only applies to the current stream. This message is only issued when you use the /LOG qualifier on the CREATE SCRIPT command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

SCRPCMDTOOLONG, script command is too long; it exceeds *count* characters

Facility: VDE, OpenVMS Development Environment

Explanation: You have entered a script command that is too long for VDE's internal script command buffer. The message indicates the maximum size of this buffer. The total length of the script command includes the initial line for the command and all continuation lines.

User Action: Change the script command so that it is not so long.

SCRPDEL, script deleted from the database script *script*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated script for the current stream has been deleted from the VDE database. This message

is only issued when you use the /LOG qualifier on the DELETE SCRIPT command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

SCRPEXTR, script extracted to file *file* script *script*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated script has been extracted from the VDE database into the indicated file. This message is only issued when you use the /LOG qualifier on the EXTRACT SCRIPT command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

SCRPFACFAIL, script for facility *facility* terminated with errors

Facility: VDE, OpenVMS Development Environment

Explanation: The script for the indicated facility terminated with errors when executed for the current command. The type of the script depends on the command. For the CREATE STREAM command, for example, the new-stream script is invoked to perform part of the stream creation.

User Action: Correct the error in the script and reenter the command.

SCRPFILE, script produced DCL command file *file* script *script*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated script has been invoked to produce the DCL command file whose name is given in the message. This message is only issued when you use the /LOG qualifier on the INVOKE SCRIPT command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

SCRPFUNCUNDEF, script function *function* is undefined in the current script

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated script function is used in the currently executing script, but is undefined for this type of script. This means that the function is not allowed to be used in this type of script. The module-name functions, for example, are undefined in scripts invoked at stream creation because these scripts are not applied to individual modules.

User Action: Do not use the indicated function in the script.

SCRPMOD, script information modified in the database script *script*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that information for the indicated script has been modified in the VDE database. The modified attributes only apply to the current stream. This message is only issued when you use the /LOG qualifier on the MODIFY SCRIPT command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

SCRPMODFAIL, *script* script for module *module* terminated with errors

Facility: VDE, OpenVMS Development Environment

Explanation: The script for the indicated module terminated with errors when executed for the current command. The type of the script is specified in the message.

User Action: Correct the error in the script and reenter the command.

SCRPUSED, script *stript* was used

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated script was used to generate one segment of the current DCL command file. This message is only issued when you use the /LOG qualifier on the INVOKE SCRIPT command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

SCRQUOERR, no closing quotation mark in script command:!/*cmd*

Facility: VDE, OpenVMS Development Environment

Explanation: VDE has detected a missing quote in the specified command for the script. The script command must be respecified using correct syntax.

User Action: The script command must be respecified using correct syntax.

SCRWAIT, commencing wait for script

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that VDE is commencing a wait operation for a script.

User Action: Contact VDE support for assistance, as the version of VDE in use contains debugging images.

SCTCONF, SCT note conference null; module not replaced

Facility: VDE, OpenVMS Development Environment

Explanation: The required SCT notes conference name field was found empty on a replacement operation.

User Action: Please reenter the SCT notes conference name using correct syntax, and retry the replacement operation.

SCTCONFSYN, SCT note conference syntax error; module not replaced

Facility: VDE, OpenVMS Development Environment

Explanation: VDE detected a syntax error in the SCT notes conference name field.

User Action: Please reenter the SCT notes conference name using correct syntax, and retry the replacement operation.

SCTFCMT, SCT fold comment required; module not replaced

Facility: VDE, OpenVMS Development Environment

Explanation: The required SCT fold comment field was found empty on a replacement operation.

User Action: Enter an SCT fold comment and retry the replacement operation.

SCTFNF, unable to locate notes conference *confname*

Facility: VDE, OpenVMS Development Environment

Explanation: VDE was unable to open the specified SCT notes conference

User Action: Please reenter the SCT notes conference name using the correct name and syntax, and retry the operation. If the conference is missing, contact the project leader, the system manager(s), or VDE support for assistance.

SCTNOTE, SCT note number null; module not replaced

Facility: VDE, OpenVMS Development Environment

Explanation: The required SCT note number field was found empty on a replacement operation.

User Action: Enter the number of the associated SCT note using correct syntax, and retry the replacement operation.

SCTNOTESYN, SCT note number syntax error; module not replaced

Facility: VDE, OpenVMS Development Environment

Explanation: VDE detected a syntax error in the SCT note number field.

User Action: Please reenter the SCT note number using correct syntax, and retry the replacement operation.

SECMERFAI, secondary merge failure

Facility: VDE, OpenVMS Development Environment

Explanation: internal state information for VDE$_INVDIRSPEC error.

User Action: Correct the cause of the VDE$_INVDIRSPEC error.

SEECONLIB, see the VDE CONVERT LIBRARY command

Facility: VDE, OpenVMS Development Environment

Explanation: The VDE database does not have the proper format, and the automatic conversion of the database has been disabled. The VDE CONVERT LIBRARY command can be used to force the conversion of the database format.

User Action: If use of the new version of VDE is intended, perform a full library backup using the database-provided backup tools, and then use the VDE CONVERT LIBRARY command to update the database format.

SESADDED, session *session* added to the database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated session definition has been added to the VDE database. This message is only issued when you use the /LOG qualifier on the REPLACE command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

SESBELUSER, session *session* belongs to another user

Facility: VDE, OpenVMS Development Environment

Explanation: You are attempting to modify or cancel a reservation session that belongs to another user, and you do not have the PERFREP privilege that would allow you to cancel or modify another user's session. As a result, the session is not cancelled or modified.

User Action: Reenter the command specifying your own reservation session to cancel or modify, or else have someone with the necessary privilege perform the operation.

SESCAN, session *session* cancelled

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated session has been cancelled. This message is only issued when you use the /LOG qualifier on the CANCEL SESSION command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

SESEXISTS, session *session* already exists in database; session not changed

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated session name was specified on a REPLACE command, but an entry for that session is already present in the database. Hence a new entry was not created for that session. Alternatively, the indicated name was specified as the new name for an existing session on the MODIFY SESSION command, but another session already has the requested new name. Hence the name of the existing session could not be changed to the new name.

User Action: If you entered the wrong session name, reenter the command with the correct name. If you entered the right session name, use the SHOW SESSION command to display the database information for that session. If you are not satisfied with that information, use the MODIFY SESSION command to change it.

SESMOD, information for session *session* modified in the database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that information for the indicated session has been modified in the VDE database. This message is only issued when you use the /LOG qualifier on the MODIFY SESSION command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

SESNOMODS, session *session* contains no module reservations

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the reservation session you are replacing or unreserving contains no module reservations. The command deleted the session from the VDE database, but did not replace or unreserve any modules.

User Action: If you meant to replace modules you have already reserved, replace them individually. Alternatively, use the RESERVE command to create a new session that contains reservations for the desired modules. Then change the modules and replace the session.

SESNOTFOUND, session *session* not found in database; parameter ignored

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated session name was entered on a SHOW SESSION command, but no such session was found in the VDE database.

User Action: This is not necessarily an error; the message simply tells you that the indicated session does not exist in the database.

SESTERM, session *session* terminated

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated session has been terminated. This message is only issued when you use the /SESSION and /LOG qualifiers on the REPLACE or UNRESERVE command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

SETKEY, keypad state has been set to *state-name*

Facility: VDE, OpenVMS Development Environment

Explanation: For the SET KEY operation, this informs the user that the key was set.

User Action: If you do wish to see this information, use the /NOLOG qualifier on the SET KEY command.

SETKEYERR, error in processing SET KEY command

Facility: VDE, OpenVMS Development Environment

Explanation: The screen management facility is used to maintain screen bindings. In this instance, SMG$SET_DEFAULT_STATE has returned an error.

User Action: See SMG documentation to determine the problem. If your answer cannot be found, please contact VDE support for assistance.

SETLIB, use SET LIBRARY, VDE$LIBRARY, or /LIBRARY

Facility: VDE, OpenVMS Development Environment

Explanation: VDE is unable to perform the specified operation as the target VDE library—the library on which the operation will be performed—has not been established.

User Action: Specify the target VDE library. To determine the available libraries, issue the "SET LIBRARY" command with no parameters. As part of the error message, this command will display the names of the predefined VDE libraries, the Rdb root directory for each, and the mnemonic name for each. Once the mnemnonic name or the Rdb root directory name for the target library is known, one can: A) use the mnemnonic name or the Rdb root directory name as the libname parameter on a "SET LIBRARY libname" VDE command. B) Use use the mnemnonic name or the Rdb root directory name as the parameter on the VDE /LIBRARY=libname. C) Define the logical name VDE$LIBRARY to translate to the VDE mnemnonic name or to the Rdb root directory name.

SETUNAUTH, attempt to set unauthorized privileges; privileges not set

Facility: VDE, OpenVMS Development Environment

Explanation: You have attempted to use the SET PRIVILEGES command to grant yourself one or more privileges that you are not authorized to set according to the VDE database.

User Action: Use the SHOW PRIVILEGES/FULL command to see what privileges you are authorized to set. Then reenter the SET PRIVILEGES command to set only authorized privileges. Alternatively, ask the project administrator to grant you authorization to set the privileges you want. Then exit from VDE, start a new VDE session, and set the privileges you want.

SHOKEYERR, error in processing SHOW KEY command

Facility: VDE, OpenVMS Development Environment

Explanation: The screen management facility is used to maintain screen bindings. In this instance, SMG$LIST_KEY_DEFS has returned an error.

User Action: See SMG documentation to determine the problem. If your answer cannot be found, please contact VDE support for assistance.

SHORESFUL, use SHOW RESERVATION/FULL to check reservation stream

Facility: VDE, OpenVMS Development Environment

Explanation: The SHOW RESERVATION/FULL command is used to display outstanding reservations, and can be used to locate the particular stream in which the reservation was originally requested. Certain operations require the specification of a stream via an explicit /STREAM qualifier, or via an implicit SET STREAM default stream value.

User Action: This message is informational, and should accompany one or more other error messages.

SNDJBCERR, job controller error status for queue *queue*

Facility: VDE, OpenVMS Development Environment

Explanation: You have attempted to start a build job and the VMS job controller (the $SNDJBCW system service) returned an error status when VDE submitted a build process to a system batch queue. The message that follows gives more information to explain the error.

User Action: The action to take to correct the error depends on the nature of the error. You or your system manager should correct the error and restart the build job.

SPARE_1, placeholder for VDE$_MODEXISTS (*module*) message

Facility: VDE, OpenVMS Development Environment

Explanation: The VDE$_MODEXISTS message was reclassified from an informational message to a warning message. This entry is a placeholder that prevents the condition values of subsequent errors from changing.

User Action: See VDE$_MODEXISTS. Also contact VDE support for assistance, as VDE should no longer be reporting this error message.

SPARE_2, placeholder for VDE$_INVALANS (*answer*) message

Facility: VDE, OpenVMS Development Environment

Explanation: The VDE$_INVALANS message was reclassified from an informational message to a warning message. This entry is a placeholder that prevents the condition values of subsequent errors from changing.

User Action: See VDE$_INVALANS. Also contact VDE support for assistance, as VDE should no longer be reporting this error message.

SPARE_3, was FACNOTDEL; replaced

Facility: VDE, OpenVMS Development Environment

Explanation: This message (was FACNOTDEL) has been superceded.

User Action: Contact VDE support for assistance.

SPARE_4, was STRMNOTDEL

Facility: VDE, OpenVMS Development Environment

Explanation: message superceded (was STRMNOTDEL)

User Action: Contact VDE support for assistance

SPARE_5, was FOLDNOTFOUND

Facility: VDE, OpenVMS Development Environment

Explanation: filler for error message elevated to a warning.

User Action: This message should not appear.

SPARE_6, was BADSTATUS; replaced

Facility: VDE, OpenVMS Development Environment

Explanation: This message (was BADSTATUS) has been superceded; the BADSTATUS message now returns an error severity.

User Action: The version of message file in use does not match the version of the images running. Check for a bad or old message file, and contact VDE support for assistance.

SPARE_8, was FOLDNOTFOUND

Facility: VDE, OpenVMS Development Environment

Explanation: filler for error message elevated to a warning.

User Action: This message should not appear.

SPAWNED, subprocess spawned

Facility: VDE, OpenVMS Development Environment

Explanation: Issued when the /NOWAIT qualifier is used.

User Action: None; this is an informational message only.

SPEMERFAI, specification merge failure

Facility: VDE, OpenVMS Development Environment

Explanation: internal state information for VDE$_INVDIRSPEC error.

User Action: Correct the cause of the VDE$_INVDIRSPEC error.

SRCNOTOUT, source module *module* cannot be an output from build step

Facility: VDE, OpenVMS Development Environment

Explanation: You have specified that a source module is an output from a build step. A source module can be an input to a build step, but it can by definition not be an output. The build step definition is not added to the database.

User Action: If this error occurs on a CREATE STEP or MODIFY STEP command, correct the error and reenter the command with valid output modules.

STATNOTSPEC, the kind of statistics to display was not specified

Facility: VDE, OpenVMS Development Environment

Explanation: This qualifier requires a keyword specifying what kind of statistics to display.

User Action: Reenter the command providing a keyword specifying what kind of statistics to display.

STEPNOTCRE, build step not created; module *module* not found in database

Facility: VDE, OpenVMS Development Environment

Explanation: You have specified a module that does not exist in the VDE database as the primary input to a compile step. As a result, the CREATE STEP command does not add the compile step or its dependency information to the database.

User Action: Correct the module name to be the name of an existing module or create the module with the CREATE MODULE command before reentering the CREATE STEP command.

STEPNOTFOUND, build step for module *module* not found in database

Facility: VDE, OpenVMS Development Environment

Explanation: You have specified a build step that does not exist in the VDE database. As a result, the current MODIFY STEP command cannot modify any information in the database.

User Action: Reenter the command specifying an existing build step or create the desired build step with the CREATE STEP command.

STEPNOTMOD, build step not modified; module *module* not found in database

Facility: VDE, OpenVMS Development Environment

Explanation: You have specified a module that does not exist in the VDE database as the primary input or output module for a build step. As a result, the MODIFY STEP command cannot modify database information.

User Action: Correct the module name to be the name of an existing module before reentering the MODIFY STEP command, or create the module with the CREATE MODULE command and enter the step information with the CREATE STEP command.

STEPNOTREV, build step for module *module* not found in review list

Facility: VDE, OpenVMS Development Environment

Explanation: You have specified a build step that does not exist in the current build-step review list. As a result, you cannot select the build step with the module name you specified.

User Action: Reenter the SELECT command specifying a build step that is present in the build-step review list for the current REVIEW BUILD_ JOB command. Use the DIRECTORY subcommand to determine what build steps are present in the list.

STGTRUNC, string truncated

Facility: VDE, OpenVMS Development Environment

Explanation: VDE has attempted to output a string which is longer than its internal buffer.

User Action: If it can be determined that it is not caused by a user error, please contact VDE support for assistance.

STKOVF, command procedures too deeply nested; limit is 16 levels

Facility: VDE, OpenVMS Development Environment

Explanation: VDE allows only 16 levels of nesting of command procedures.

User Action: Embed some of the command procedures within others to decrease the level of nesting to fewer than 16.

STRMADDED, stream *stream* added to the database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated stream definition has been added to the VDE database. This message is only issued when you use the /LOG qualifier on the CREATE STREAM command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

STRMCLOSED, stream *stream* is permanently closed; state not changed

Facility: VDE, OpenVMS Development Environment

Explanation: You have tried to change the state of a stream development stream with the /OPEN or /FROZEN qualifier on the MODIFY STREAM command when the stream is already in the permanently closed state. The rest of the command still takes effect, but the stream state is not changed; the stream remains closed.

User Action: No user action is necessary; the state cannot be changed and the command did not change it.

STRMDEL, stream *stream* deleted from the database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the definition for the indicated stream has been deleted from the VDE database. This message is only issued when you use the /LOG qualifier on the DELETE STREAM command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

STRMEXISTS, stream *stream* already exists in database

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated stream name was specified on a CREATE STREAM command, but an entry for that stream is already present in the database. Hence a new entry was not created for that stream. Alternatively, the indicated name was specified as the new name for an existing stream on the MODIFY STREAM command, but another stream already has the requested new name. Hence the name of the existing stream could not be changed to the new name.

User Action: If you entered the wrong stream name, reenter the command with the correct name. If you entered the right stream name, use the SHOW STREAM command to display the database information for that stream. If you are not satisfied with that information, use the MODIFY STREAM command to change it.

STRMMOD, information for stream *stream* modified in the database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that information for the indicated stream has been modified in the VDE database. This message is only issued when you use the /LOG qualifier on the MODIFY STREAM command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

STRMNOTCLOSED, stream *stream* can not be deleted because it is not closed

Facility: VDE, OpenVMS Development Environment

Explanation: You are trying to delete a stream which is in an open or frozen state. Only streams in a closed state may be deleted.

User Action: If appropriate, change the state of the stream to closed and repeat the delete stream command.

STRMNOTDEL, stream *stream* not deleted

Facility: VDE, OpenVMS Development Environment

Explanation: You are attempting to delete a stream that cannot be deleted. Additional messages are normally included with this message, and these messages indicate why the strean cannot be deleted.

User Action: Resolve the problem(s) flagged by the other error messages, and reenter the DELETE STREAM command.

STRMNOTFOUND, stream *stream* not found in database; parameter ignored

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated stream name was entered on a SHOW STREAM command, but no such stream was found in the VDE database.

User Action: This is not necessarily an error; the message simply tells you that the indicated stream does not exist in the database.

STRMNOTOPEN, stream *stream* is not open; operation not allowed

Facility: VDE, OpenVMS Development Environment

Explanation: You are trying to reserve or replace a module, but the currently active stream (whose name is given in the message) is either frozen or closed. It is thus not open for the reservation or replacement of modules.

User Action: You can use the SHOW STREAM/FULL command to see the status of the current stream. If the stream is closed, it is permanently protected against being changed, and you should not be trying to reserve or replace modules in it. If the stream is frozen, it is currently protected against being changed, but changes may be allowed later when the library administrator sets the state back to open. You must then wait until the stream becomes open. (You may reserve modules from a frozen stream, but not replace them.) Finally, if you meant to reserve or replace modules in another stream, you should use the SET STREAM command to switch your context to that stream and then reserve or replace the desired modules.

STRMNOTPOP, stream *stream* is not allowed to be populated

Facility: VDE, OpenVMS Development Environment

Explanation: You are attempting to populate the indicated stream with the POPULATE STREAM command but a stream attribute disallows use of this command for the stream. Hence the stream has not been populated.

User Action: Either you should not be trying to populate the stream or else you must set the stream attribute that allows you to populate the stream before retrying the operation. To set that attribute, use the /POPULATE qualifier to the MODIFY STREAM command.

STRMPOP, stream *stream* populated from stream *stream*

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated stream has been populated with all generations from the second indicated stream. This message is only issued when you use the /LOG qualifier on the POPULATE STREAM command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

STRMPOPCOUNT, populate module counts: update *count*, insert *count*, remove *count*

Facility: VDE, OpenVMS Development Environment

Explanation: This message tells you how many modules were updated in, inserted into, and removed from the target stream by the current POPULATE STREAM command. This message is only issued when you use the /LOG qualifier on the POPULATE STREAM command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

STRTOOLONG, quoted string too long; must be shortened

Facility: VDE, OpenVMS Development Environment

Explanation: The maximum length of the quoted string is determined by the nature of the string. For the DEFINE command, the string is limited to 255 characters. For remark strings, it is limited to 132 characters.

User Action: Try to shorten the size of the string.

SUCGENEXIST, successor generation for *module* already exists in library

Facility: VDE, OpenVMS Development Environment

Explanation: The new generation of the indicated module created by the current REPLACE or PERFORM REPLACEMENT command should get a whole-number CMS generation expression but the reserved generation already has a successor generation in the VDE library with that whole-number CMS generation expression. This situation can arise if you have placed the reserved generation into a stream using the INSERT GENERATION command or if you have changed the stream that

‘‘owns’’ the whole-number successor generation with the /MAIN_STREAM qualifier to the MODIFY GENERATION command.

User Action: If the existing whole-number successor generation should not be in the VDE library, delete it and its successors with the DELETE GENERATION command. Then repeat the failed replacement. Otherwise, use the /MAIN_STREAM qualifier to the MODIFY GENERATION command to allocate ownership of the whole-number successor generation to the stream that already owns that generation, but not to any stream into which you are trying to replace the module. This will force the replacement to use a CMS variant letter. Then repeat the replacement.

SUCLNKADD, successor link from stream *stream* to stream *stream* added

Facility: VDE, OpenVMS Development Environment

Explanation: The successor stream relationship, from the first stream specified to the second stream specified, has been added.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

SUCLNKREM, successor link from stream *stream* to stream *stream* removed

Facility: VDE, OpenVMS Development Environment

Explanation: The successor stream relationship, from the first stream specified to the second stream specified, has been removed.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

SYNTAX, command syntax error at or near: ’*text*’

Facility: VDE, OpenVMS Development Environment

Explanation: All commands have a specific syntax that must be followed. Failure to do so will result in this error.

User Action: See VDE Reference Manual for assistance with the command syntax.

TOOMANPARM, too many poundsign parameters in history or notes stringinvalid string is: *string*

Facility: VDE, OpenVMS Development Environment

Explanation: The history or notes string specified contains more than one history or notes parameter. This is not allowed.

User Action: Reenter the command with a valid history or notes string.

TRANTOODEEP, translation of logical name *logical* more than 7 deep

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated logical name is either too deeply nested, or is defined incorrectly to point back to itself causing an infinite translation loop.

User Action: Reenter the command without using the logical name.

UIDSKEW, possible UID skew detected; check for correct UID file.

Facility: VDE, OpenVMS Development Environment

Explanation: VDE has detected an internal inconsistency with the UID file.

User Action: Please determine if the UID file associated with the VDE distribution is the version in use. If the UID file is the one associated with the distribution, please contact contact VDE support for assistance. Please include any information that may be useful or relevent in duplicating the problem. If there is a skew code associated with this error, please include it in the error report.

UIDSKEWCODE, UID skew code *value*

Facility: VDE, OpenVMS Development Environment

Explanation: This is the code for the particular internal inconsistency detected within the UID file.

User Action: Please determine if the UID file associated with the VDE distribution is the version in use. If the UID file is the one associated with the distribution, please contact contact VDE support for assistance.

UNACLOFIL, unable to close file *file-name*

Facility: VDE, OpenVMS Development Environment

Explanation: An error occurred when trying to close the indicated text file. A more specific error message follows this message.

User Action: Examine the error message that follows to diagnose the reason for the error. Then correct the problem accordingly.

UNACLOLOG, unable to close log file *file-name*

Facility: VDE, OpenVMS Development Environment

Explanation: The SYS$CLOSE system service was not performed successfully in closing the log file.

User Action: Determine what could be preventing RMS from closing the file successfully. If the reason cannot be determined, please contact VDE support for assistance.

UNACONLOG, unable to connect log file *file-name*

Facility: VDE, OpenVMS Development Environment

Explanation: The SYS$CONNECT system service was not performed successfully in connecting the log file's RAB and FAB.

User Action: Determine what could be preventing RMS from connecting the file's FAB and RAB successfully. If this cannot be done, please contact VDE support for assistance.

UNACONVDBFORM, unable to convert database format due to access conflict with another user

Facility: VDE, OpenVMS Development Environment

Explanation: The VDE database does not have the format required by the current version of VDE and VDE has attempted to convert the database to the required format. This operation has failed because of an access conflict with another user attempting to convert the database at the same time. That other user has probably succeeded in converting the database format, which is why you cannot do so.

User Action: Retry the CONVERT LIBRARY command and try accessing the database again. You should find that you are now able to convert the database format or that the format has already been converted. Either way, you should be able to use VDE with that database.

UNACREFIL, unable to create file *file-name*

Facility: VDE, OpenVMS Development Environment

Explanation: An error occurred when trying to create or open a new text file. A more specific error message follows this message.

User Action: Examine the error message that follows to diagnose the reason for the error. Then correct the problem accordingly.

UNACRELOG, unable to create log file *file-name*

Facility: VDE, OpenVMS Development Environment

Explanation: The SYS$CREATE system service was not performed successfully in creating the output file.

User Action: Determine what could be preventing RMS from creating the file successfully. If the reason cannot be determined, please contact VDE support for assistance.

UNACREMBX, unable to create temporary mailbox

Facility: VDE, OpenVMS Development Environment

Explanation: An error occurred when trying to create a mailbox for sending text between processes. A more specific error message follows this message.

User Action: Examine the error message that follows to diagnose the reason for the error. Then correct the problem accordingly.

UNADELFIL, unable to delete file *file-name*

Facility: VDE, OpenVMS Development Environment

Explanation: An error occurred when trying to delete the indicated file. A more specific error message follows this message.

User Action: Examine the error message that follows to diagnose the reason for the error. Then correct the problem accordingly.

UNAOPEFIL, unable to open file *file-name*

Facility: VDE, OpenVMS Development Environment

Explanation: An error occurred when trying to open an existing text file for reading. A more specific error message follows this message.

User Action: Examine the error message that follows to diagnose the reason for the error. Then correct the problem accordingly.

UNAOPEINP, unable to open input stream '*text*'

Facility: VDE, OpenVMS Development Environment

Explanation: VDE could not read the command line correctly.

User Action: The logical name VDE$INPUT is probably not defined correctly. Correct the VDE$INPUT reference.

UNAPARFIL, unable to parse file name *name* with default string *name*

Facility: VDE, OpenVMS Development Environment

Explanation: An error occurred when trying to parse the indicated file specification with the indicated default name string. A more specific error message follows this message.

User Action: Examine the error message that follows to diagnose the reason for the error. Then correct the problem accordingly.

UNAREPNOTECONF, unable to reply to note in notes conference *conference*

Facility: VDE, OpenVMS Development Environment

Explanation: VDE attempted to record a PERFORM REPLACEMENT operation by replying to a previous note for the REPLACE command in the indicated notes conference, but was unable to find the notes conference or the note to reply to. As a result, the PERFORM REPLACEMENT operation was not recorded in the notes conference.

User Action: Make sure that the notes conference has been established and that the notes created by the REPLACE command are not being deleted.

UNDKEY, *state-name* key *key-name* is undefined

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs the user that the key was undefined by either the DELETE/KEY or UNDEFINE/KEY commands.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

UNKCOMSTA, unknown completion status value *status* passed to VDESTATUS

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the image VDESTATUS, was passed a status value it did not understand. VDESTATUS is expecting the character string "SUCCESS" or the character string "FAILURE" to be passed to it for its parameter. Any other character string results in this error message being output.

User Action: Check the invocation of the VDESTATUS image. Typically, VDESTATUS is invoked from a command file that is built by the program VDEDOBUILD for running buildsteps. Check the command file associated with the buildstep that resulted in this message.

UNKEPT, unable to *opnam* to kept subprocess section

Facility: VDE, OpenVMS Development Environment

Explanation: VDE has encountered an error while attempting to create or connect to the global section used to communicate with the kept subprocess.

User Action: Attempt to resolve the problem based on any other error messages included with this one, check the documentation for information on setting up process and system quotas, and contact VDE support for assistance.

UNKEYW, unrecognized keyword '*text*' - check validity and spelling

Facility: VDE, OpenVMS Development Environment

Explanation: The keyword provided in the command was unrecognized in this context.

User Action: See VDE Reference Manual for assistance with the command syntax.

UNKFILEXT, Unknown file extension in analysis file; the complete filespecification found was *spec*

Facility: VDE, OpenVMS Development Environment

Explanation: The program VDEDEPEND encountered a file specification that has a file extension it does not support. This error occurred while VDEDEPEND was processing an analysis (.ANA) file recording compiler or assembler dependency information.

User Action: Determine if the analysis file is corrupted. If the analysis file is not corrupted and the file extension corresponds to a file type that should be recorded as dependency information, please contact VDE support for assistance.

UNKPNAM, unable to locate project *projnam* in stream *streamnam*

Facility: VDE, OpenVMS Development Environment

Explanation: VDE has unable to locate the specified project name associated with the specified stream.

User Action: Specify a valid and correctly-spelled project name from the available list of projects assocated with the stream. If you believe that there should be one or more additional projects associated with the stream, please contact the project leader, the system manager(s), or VDE support for assistance.

UNKPROJ, unable to locate a project name

Facility: VDE, OpenVMS Development Environment

Explanation: VDE has encountered an error while using the notes callable interface.

User Action: Attempt to resolve the problem based on any other error messages included with this one, check the documentation for information on setting up the notes conferences, and contact VDE support for assistance.

UNKQARCLD, internal VDE coding error

Facility: VDE, OpenVMS Development Environment

Explanation: VDE has detected an internal inconsistency.

User Action: Please contact VDE support for assistance. Please include any information that may be useful or relevent in duplicating the problem.

UNLOCKED, overriding library lockout

Facility: VDE, OpenVMS Development Environment

Explanation: This message is displayed when user with MODLIB privilege replaces a module in a locked library.

User Action: Evaluate if the replacement is necessary in the locked library; unlock the library if appropriate.

UNMATQUO, unmatched quote or missing quote at end of string

Facility: VDE, OpenVMS Development Environment

Explanation: Any token that is to be delimited by quotes must have quotes on both sides of the token. These quotes may be either single or double; however, the delimiters must be the same characters.

User Action: Reenter the command with the appropriate quotes balanced.

UNRESERVED, reservation for module *module* cancelled

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated module has been unreserved in the current library and is therefore no longer reserved. This message is only issued when you use the /LOG qualifier on the UNRESERVE command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

USEDELMOD, to unreserve a CREATE MODULE reservation, use DELETE MODULE

Facility: VDE, OpenVMS Development Environment

Explanation: An erroneous attempt was made to UNRESERVE a just-created module. To cancel the module reservation and the module creation caused by a CREATE MODULE command, use the DELETE MODULE command. After the DELETE MODULE command is accepted, the module will not be created, the outstanding module reservation will be canceled, and no further user action is required.

User Action: If you wish to cancel this reservation, use the DELETE MODULE command. This message is informational, and should accompany one or more other error messages.

USERADDED, user *user* added to the database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated user has been added to the VDE database. This message is only issued when you use the /LOG qualifier on the CREATE USER command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

USERDEL, user *user* deleted from the database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated user has been deleted from the VDE database. This message is only issued when you use the /LOG qualifier on the DELETE USER command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

USEREXISTS, user *user* already exists in database; user not changed

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated username was specified on a CREATE USER command, but an entry for that user is already present in the database. Hence a new entry was not created for that user. Alternatively, the indicated username was specified as the new username for an existing user on the MODIFY USER command, but another user already has the

requested new username. Hence the username of the existing user could not be changed to the new username.

User Action: If you entered the wrong username, reenter the command with the correct username. If you entered the right username, use the SHOW USER command to display the database information for that user. If you are not satisfied with that information, use the MODIFY USER command to change it.

USERMOD, information for user *user* modified in the database

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that information for the indicated user has been modified in the VDE database. This message is only issued when you use the /LOG qualifier on the MODIFY USER command.

User Action: Use the /NOLOG qualifier for the command if you do not wish to see the message.

USERNOTDEL, user *user* not deleted; you cannot delete yourself

Facility: VDE, OpenVMS Development Environment

Explanation: This message informs you that the indicated user, namely you yourself, has not been deleted from the database. You are not allowed to delete yourself from the database. This prohibition prevents you from accidentally deleting yourself from the database, especially when using a wildcarded username specification. There is thus no way to delete all users from the database.

User Action: If you want to delete yourself from the database, someone else with the required privilege must do it for you.

USERNOTFOUND, user *user* not found in database; parameter ignored

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated username was entered on a SHOW USER command, but no such user was found in the VDE database.

User Action: This is not necessarily an error; the message simply tells you that the indicated user does not exist in the database.

USERNOTUAF, user *user* not found in system User Authorization File

Facility: VDE, OpenVMS Development Environment

Explanation: The indicated username is your username, but it was not found in the system User Authorization File.

User Action: This error should not happen except to VDE developers who redefine the current user during testing. Submit a problem report.

USEVERSLNM, use logical name VDE$VMS_VERSION to override OpenVMS version

Facility: VDE, OpenVMS Development Environment

Explanation: VDE is unable to determine OpenVMS version information from the version string returned from a SYS$GETSYI service call. This message recommends that you use the VDE$VMS_VERSION logical name to override the system value.

User Action: Define the logical name VDE$VMS_VERSION to a valid OpenVMS version string value.

VALREQ, missing qualifier or keyword value - supply all required values '*text*'

Facility: VDE, OpenVMS Development Environment

Explanation: In many commands, specific keywords or qualifiers are expected.

User Action: See VDE Reference Manual for assistance with the command syntax.

VDECLUSTER, vde.cluster: *cluster*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

VDECONTACT, vde.contact: please contact VDE support for assistance

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

VDEDATETIME, vde.datetime: *datetime*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

VDEFORMAT, vde.format: the format of this file is subject to change

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

VDENAME, vde.product: *vdename*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

VDENODE, vde.node: *node*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

VDEPROJECT, vde.project: *project*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

VDERDBBKP, VDE$DATABASE$*db*_BCKP_*type*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

VDERDBBKP_FULL, FULL

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

VDERDBBKP_INCR, INCR

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

VDEVERSION, vde.version: *version*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to format text strings within VDE, and should not be visible to users as an error message.

User Action: Contact VDE support for assistance, as the version of VDE in use should not be displaying this as an error message.

VERBOTEN, direct ATTACH to a VDE kept subprocess is forbidden

Facility: VDE, OpenVMS Development Environment

Explanation: An attempt has been made to directly ATTACH to a VDE kept subprocess via a DCL command, and this is not permitted.

User Action: Use of the DCL ATTACH command to attach to a VDE kept subprocess is not permitted. If this message arises from a condition other than an attempt by a user to ATTACH to a VDE kept subprocess, please contact VDE support for assistance. Please include any information that may be useful or relevent in duplicating the problem.

WHEREIS, component *comp* is *value*

Facility: VDE, OpenVMS Development Environment

Explanation: This message is used to display information about parts of the local VDE environment and VDE configuration.

User Action: No action is required.

WHYOFF, reason: !AD

Facility: VDE, OpenVMS Development Environment

Explanation: This message is usually paired with VDE$_OFFLINE. The VDE environment has been taken offline, and this message indicates why.

User Action: If the problem persists, please contact VDE support for assistance.

WILDGENNUM, wildcarded generation number not allowed in *module*

Facility: VDE, OpenVMS Development Environment

Explanation: You have specified a wildcarded generation number on a module name for a command which does not allow the generation number to be wildcarded.

User Action: Decide what generation of the module you want to apply the command to. Then reissue the command, specifying that generation number. Alternatively, you may omit the generation number, in which case the command is applied to the last (most recent) generation of the module.

WILDNOTALL, wildcard characters not allowed in name: *name*

Facility: VDE, OpenVMS Development Environment

Explanation: You have specified a wildcard character (asterisk or percent sign) in a name that is not allowed to have wildcard characters. The message displays the name up to and including the wildcard character.

User Action: Specify the name without a wildcard character and resubmit the command.

XMERROR, Motif interface error detected

Facility: VDE, OpenVMS Development Environment

Explanation: VDE has detected an error in the Motif interface.

User Action: A problem has occurred accessing the X Windows display, during the fetch of a widget from a UID file, or another similar error. Additional error messages should have been displayed prior to this particular error and the other messages should detail the particular failure detected. If unable to determine the case of the error, please contact VDE support for assistance. Please include any information that may be useful or relevent in duplicating the problem.

Index

A

Abbreviations
  using • 1–3
ACCEPT REPLACEMENT command • VDE–3, VDE–22
ACCEPT Review Replacement Command • VDE–257
ACE • VDE–38
At-sign (@) command • VDE–2
ATTACH command • VDE–6
ATTACH Review Build Job Command • VDE–251
ATTACH Review Replacement Command • VDE–258

B

BACK Review Build Job Command • VDE–251
BACK Review Replacement Command • VDE–257
Batch Queue • VDE–11, VDE–94, VDE–164, VDE–212, VDE–375
BUILD command • VDE–8
Build dependency recording
  Callable interface • B–2

C

CANCEL DIRECTORY command • VDE–15
CANCEL FOLD command • VDE–18
CANCEL LOG command • VDE–20
CANCEL PRIVILEGES command • VDE–21
CANCEL REPLACEMENT command • VDE–22, VDE–125
CANCEL SESSION command • VDE–24
CANCEL VERIFY command • VDE–26
CMS • VDE–40
  Bulk Upload • D–1
  Callable • VDE–44, VDE–148, VDE–242
CMS libraries • VDE–15, VDE–38, VDE–39, VDE–40, VDE–42, VDE–44, VDE–45, VDE–46, VDE–65, VDE–84, VDE–85, VDE–90, VDE–91, VDE–114, VDE–209, VDE–210, VDE–279, VDE–281, VDE–310, VDE–332, VDE–384, VDE–385, VDE–386
Commands
  Ambiguous • 1–3
Comment Character • VDE–260
COMPILE command • VDE–27
Continuation Character • VDE–260
Conversion
  Automatic • VDE–29, VDE–56, VDE–182, VDE–289
CONVERT LIBRARY command • VDE–29, VDE–56, VDE–182, VDE–289
COPY command • VDE–31
CREATE ARCHITECTURE command • VDE–33
CREATE CONTEXT command • VDE–35
CREATE DIRECTORY_TREE command • VDE–38
CREATE FACILITY/FROM_CMS_LIBRARY command • VDE–94
CREATE FACILITY command • VDE–44
CREATE FOLD command • VDE–48
CREATE GROUP command • VDE–51
CREATE LIBRARY command • VDE–29, VDE–54, VDE–182
CREATE MODULE command • VDE–65
CREATE REPLACEMENT command • VDE–73, VDE–236
CREATE SCRIPT command • VDE–75, VDE–271
CREATE SESSION command • VDE–82
CREATE STEP command • VDE–84
CREATE STREAM command • VDE–89
CREATE USER command • VDE–97
Creating
  a subprocess • VDE–370
CURRENT Review Build Job Command • VDE–251
CURRENT Review Replacement Command • VDE–257
%CURRENT Script Keyword • VDE–80, VDE–266, VDE–268, VDE–270, VDE–353

D

Database Directory • VDE–54
DCL
  DIFFERENCES Command • VDE–138
DCL Command Procedures • VDE–2
Debugging Scripts • VDE–264
Defect tracking • VDE–175, VDE–232, VDE–233, VDE–235, VDE–236
DEFINE/COMMAND command • VDE–100
DEFINE/KEY command • VDE–102
Defining keys • VDE–102
DELETE/COMMAND command • VDE–106
DELETE/KEY command • VDE–107
DELETE ARCHITECTURE command • VDE–109
DELETE BUILD_JOB command • VDE–110
DELETE CONTEXT command • VDE–112
DELETE FACILITY command • VDE–114
DELETE FOLD command • VDE–116
DELETE GENERATION command • VDE–118
DELETE GROUP command • VDE–120
DELETE MODULE command • VDE–66, VDE–123
DELETE REPLACEMENT command • VDE–125
DELETE SCRIPT command • VDE–127
DELETE STEP command • VDE–130
DELETE STREAM command • VDE–132
DELETE USER command • VDE–134
Dependency Code
  VDE$K_DEPEND_BEGIN_RECORD • B–2
  VDE$K_DEPEND_END_RECORD • B–2
  VDE$K_DEPEND_INPUT_FILE • B–2
  VDE$K_DEPEND_INPUT_LIBRARY • B–2
  VDE$K_DEPEND_INPUT_MODULE • B–2
  VDE$K_DEPEND_OUTPUT_FILE • B–2
  VDE$K_DEPEND_OUTPUT_LIBRARY • B–2
  VDE$K_DEPEND_OUTPUT_MODULE • B–2
  VDE$K_DEPEND_VERSION • B–2
Derived Module
  Build Input • VDE–8
  Change Levels • VDE–358
  Compilation • VDE–27
  Copying • VDE–31
  Defining Module As • VDE–67
  Definition • VDE–65
  Display Of • VDE–322, VDE–337
  Facility Creation • VDE–44
  Grouping • VDE–51
  Storage Of • VDE–91, VDE–332
DIFFERENCES command • VDE–137
DIRECTORY Review Build Job Command • VDE–250
DIRECTORY Review Replacement Command • VDE–255
Directory symbol • VDE–16, VDE–17, VDE–69, VDE–260, VDE–261, VDE–268, VDE–270, VDE–280, VDE–281, VDE–310
$DIR Script Function • VDE–268

E

Editor • VDE–193, VDE–233, VDE–257, VDE–284
EDIT Review Replacement Command • VDE–256
EMIT Script Command • VDE–260
Entering commands
  from DCL • 1–2, DCL–3
  within VDE • 1–2
  within VSC • 1–2
Entering subcommands
  within VDE • DCL–3
Error Messages • E–1
EXIT command • VDE–142
EXIT Review Build Job Command • VDE–251
EXIT Review Replacement Command • VDE–258
EXIT Script Command • VDE–260
EXTRACT SCRIPT command • VDE–143

F

$FACMODTYP Script Function • VDE–267
$FACROOT Script Function • VDE–268
$FAC Script Function • VDE–267
FETCH command • VDE–146
FETCH Script Command • VDE–260
File
  VDE$RECDEP • B–7
$FILEDIR Script Function • VDE–267
File Extension
  Log • VDE–291
  RDA • VDE–54, VDE–58
  RDB • VDE–54
  SNP • VDE–54, VDE–58, VDE–61
  VDE • VDE–2
  VDE$COMMENT • VDE–4, VDE–225
  VDE$LOG • VDE–255
  VDESCRIPT • VDE–75, VDE–143
$FILESPEC Script Function • VDE–267
FIRST Review Build Job Command • VDE–251

FIRST Review Replacement Command • VDE–257

H

HELP command • VDE–151
  example • 1–3
History
  CMS Element • VDE–57, VDE–58, VDE–59, VDE–67, VDE–147, VDE–149, VDE–183, VDE–188, VDE–241
  Displaying • VDE–321
  Modules • VDE–321
  Notes • VDE–58, VDE–183

I

INSERT GENERATION command • VDE–152
INVOKE SCRIPT command • VDE–155
INVOKE SCRIPT Command • VDE–264

K

Kept Subprocess • VDE–2
Key definitions
  displaying • VDE–329

L

LAST Review Build Job Command • VDE–251
LAST Review Replacement Command • VDE–257
LIB$GET_VM • VDE–335
LIB$GET_VM_PAGE • VDE–335
LIB$REC_DEPENDENCY • B–2
Libraries
  Labels • A–1
  Mnemonics • A–2
  Names • A–1
$LIBROOT Script Function • VDE–268
LINK command • VDE–77, VDE–159
LINKER • B–1
Logical name
  VDE$LIBRARY_DEFAULTS_LABELS • VDE–184
  VDE$LIBRARY_DEFAULTS_MNEMONICS • VDE–184
Logical name (cont'd)
  VDE$LIBRARY_LABEL • VDE–184
  VDE$LIBRARY_MNEMONIC • VDE–184
Logical Name
  Copying • VDE–370, VDE–372
  Macro Assembler • VDE–263
  SYS$LOGIN • VDE–138
  SYS$OUTPUT • VDE–370
  SYS$SCRATCH • VDE–138, A–3
  VDE$DB_AREA • VDE–54
  VDE$LIBRARY • DCL–3, DCL–4, DCL–5, DCL–9, VDE–54, VDE–55, VDE–62, VDE–289, A–1
  VDE$LIBRARY_DEFAULTS • VDE–290, A–1
  VDE$LIBRARY_DEFAULTS_CMS • A–1
  VDE$LIBRARY_DEFAULTS_LABELS • VDE–290, A–1
  VDE$LIBRARY_DEFAULTS_MNEMONICS • VDE–290, A–1
  VDE$LIBRARY_LABEL • A–2
  VDE$LIBRARY_MNEMONIC • A–2
  VDE$NOTES_LIBRARY • VDE–90, VDE–210, A–2
  VDE$OFFLINE • A–2
  VDE$PAGEFILE_SECTION • B–7
  VDE$RECORD_DEPENDENCY • B–7
  VDE$REPLACE_DEFAULT • A–2
  VDE$SCRATCH • VDE–138, A–3
  VDE$SNAPSHOT • VDE–54
  VDE$TEMPLATES • A–3
  VDE$WHEREIS_BACKUPS_AFTER_IMAGE • DCL–11, A–3
  VDE$WHEREIS_BACKUPS_ARCHIVE • DCL–11, A–3
  VDE$WHEREIS_BACKUPS_CURRENT • DCL–11, A–3
  VDE$WHEREIS_BACKUP_FULL • DCL–12, A–3
  VDE$WHEREIS_BACKUP_INCR • DCL–11, A–3
  VDE$WHEREIS_BUILD • DCL–12, A–3
  VDE$WHEREIS_CMS • DCL–11, A–3
  VDE$WHEREIS_DATABASE • DCL–11, A–3
  VDE$WHEREIS_LABEL • DCL–11, A–3
  VDE$WHEREIS_MNEMONIC • DCL–11, A–3
  VDE$WHEREIS_STAGING • DCL–12, A–4
  VDE$WHEREIS_STATISTICS • DCL–12, A–4

M

Mail
  Lost Reservation Notification • VDE–118, VDE–153, VDE–227, VDE–230

Mail (cont'd)
Notification • VDE–57, VDE–60, VDE–69, VDE–93, VDE–98, VDE–183, VDE–185, VDE–189, VDE–211, VDE–216
Queued Reservation Notification • VDE–244
Reviewer Notification • VDE–70, VDE–95, VDE–98, VDE–213, VDE–216, VDE–236
Mainline
marking a file as • VDE–92, VDE–173, VDE–187
Main Stream
See also Mainline
Marker file • VDE–66, VDE–69, VDE–188, VDE–189, VDE–281
Messages • E–1
MODIFY ARCHITECTURE command • VDE–161
MODIFY BUILD_JOB command • VDE–163
MODIFY CONTEXT command • VDE–166
MODIFY FACILITY /DELETE command • VDE–114
MODIFY FACILITY command • VDE–169
MODIFY FOLD command • VDE–171
MODIFY GENERATION/MAIN_STREAM command • VDE–92, VDE–187
MODIFY GENERATION/OBSOLETE command • VDE–187
MODIFY GENERATION/REMARK command • VDE–187, VDE–190
MODIFY GENERATION command • VDE–173
MODIFY GROUP command • VDE–177
specifying command qualifiers for • VDE–178
specifying parameter qualifiers for • VDE–178
MODIFY LIBRARY/STATISTICS command • VDE–232, VDE–235
MODIFY LIBRARY command • VDE–181, VDE–289
MODIFY MODULE command • VDE–187
MODIFY REPLACEMENT command • VDE–192
MODIFY RESERVATION command • VDE–195
MODIFY SCRIPT command • VDE–200
MODIFY SESSION command • VDE–203
MODIFY STEP command • VDE–205
MODIFY STREAM command • VDE–209
MODIFY USER/USER command • VDE–135
MODIFY USER command • VDE–215
$MOD Script Function • VDE–268
$MODTYP Script Function • VDE–269
Motif • DCL–3, DCL–15

N

NEXT Review Build Job Command • VDE–251
NEXT Review Replacement Command • VDE–257
Notes • VDE–230
Notification • VDE–56, VDE–59, VDE–90, VDE–182, VDE–209

O

Obsolete
marking a file as • VDE–173, VDE–187

P

%PARENT Script Keyword • VDE–80, VDE–266, VDE–268, VDE–270, VDE–353
PERFORM REPLACEMENT command • VDE–22, VDE–218, VDE–234
POPULATE STREAM command • VDE–221
Privilege
BUILD • VDE–8, VDE–27, VDE–31, VDE–110, VDE–159, VDE–163, VDE–186, VDE–217, VDE–373, VDE–376, VDE–379
BYPASS • DCL–9, DCL–10, VDE–289, VDE–294, VDE–295
CREARCH • VDE–33, VDE–161
CRECTX • VDE–35, VDE–36, VDE–112, VDE–113, VDE–166, VDE–167
CREFAC • VDE–44, VDE–99, VDE–169
CREGRP • VDE–51, VDE–120, VDE–177
CREMOD • VDE–65, VDE–66, VDE–123, VDE–186, VDE–187
CRESCRP • VDE–75, VDE–127, VDE–200, VDE–259, VDE–267, VDE–271
CRESTEP • VDE–84, VDE–130, VDE–205
CRESTRM • VDE–15, VDE–38, VDE–89, VDE–209, VDE–210, VDE–211, VDE–212, VDE–213, VDE–278
CREUSR • VDE–97, VDE–134, VDE–215
DELARCH • VDE–109
DELFAC • VDE–114
DELMOD • VDE–66, VDE–123
DELREP • VDE–125
DELSTRM • VDE–132
FROZEN • VDE–21, VDE–93, VDE–212, VDE–222
MODLIB • VDE–29, VDE–61, VDE–181, VDE–184
MODSTRM • VDE–209
Override • VDE–289, VDE–294, VDE–295

Privilege (cont'd)
PERFREP • VDE–24, VDE–67, VDE–68, VDE–118, VDE–125, VDE–152, VDE–192, VDE–203, VDE–218, VDE–221, VDE–227, VDE–386
PREFREP • VDE–22
READALL • DCL–9
RESREP • VDE–3, VDE–18, VDE–22, VDE–24, VDE–48, VDE–73, VDE–82, VDE–116, VDE–125, VDE–171, VDE–173, VDE–192, VDE–195, VDE–203, VDE–224, VDE–229, VDE–239, VDE–381
UNCLOSE • VDE–93, VDE–212
USERNAME • VDE–5, VDE–19, VDE–21, VDE–50, VDE–117, VDE–172, VDE–195, VDE–197, VDE–226, VDE–230, VDE–237, VDE–240, VDE–245, VDE–381, VDE–383
Process
attaching to another • VDE–6
Project tracking • VDE–234
Prompt
CLD • VDE–234
Customizing • VDE–296
Library Name • VDE–59
Name • VDE–57, VDE–98
QAR • VDE–234
Remark • VDE–49, VDE–59, VDE–70, VDE–73, VDE–83, VDE–94, VDE–98, VDE–225, VDE–226, VDE–237, VDE–257
Root Directory • VDE–54
Stream • VDE–49, VDE–119, VDE–149, VDE–154, VDE–228, VDE–237, VDE–245
VDEREVBLD • VDE–247, VDE–248, VDE–250
VDEREVREP • VDE–252
Protections
Directory • VDE–38

Q

Queue • VDE–11, VDE–94, VDE–164, VDE–212, VDE–375
Queued Replacement • VDE–3, VDE–15, VDE–22, VDE–39, VDE–55, VDE–68, VDE–70, VDE–73, VDE–79, VDE–90, VDE–94, VDE–95, VDE–125, VDE–126, VDE–181, VDE–191, VDE–192, VDE–193, VDE–209, VDE–213, VDE–217, VDE–218, VDE–219, VDE–224, VDE–232, VDE–235, VDE–252, VDE–253, VDE–270, VDE–278, VDE–280, VDE–284, VDE–310, VDE–342, VDE–385
Quoting characters
EMIT script command • VDE–260

R

REJECT REPLACEMENT command • VDE–22, VDE–224
REJECT Review Replacement Command • VDE–257
REMOVE GENERATION command • VDE–227
REPLACE command • VDE–22, VDE–229, A–3
RESERVE command • VDE–239
Resource File • A–1
Review Build Commands • VDE–250
Review Build Job Command
ATTACH • VDE–251
BACK • VDE–251
CURRENT • VDE–251
DIRECTORY • VDE–250
EXIT • VDE–251
FIRST • VDE–251
LAST • VDE–251
NEXT • VDE–251
SELECT • VDE–251
SPAWN • VDE–251
REVIEW BUILD_JOB command • VDE–247
Review Replacement Command
ACCEPT • VDE–257
ATTACH • VDE–258
BACK • VDE–257
CURRENT • VDE–257
DIRECTORY • VDE–255
EDIT • VDE–256
EXIT • VDE–258
FIRST • VDE–257
LAST • VDE–257
NEXT • VDE–257
REJECT • VDE–257
SELECT • VDE–255
SHOW • VDE–256
SPAWN • VDE–257

REVIEW REPLACEMENT command • VDE–252
Review Replacement Commands • VDE–255

S

Scratch Directory • VDE–138, VDE–139, VDE–155, A–3
$SCRATCH Script Function • VDE–269
Script Command • VDE–259
  EMIT • VDE–260
  EMIT quoting character pairs • VDE–260
  EXIT • VDE–260
  FETCH • VDE–260
Script Comment Character • VDE–260
Script Continuation Character • VDE–260
Script Debugging • VDE–264
Script Function
  $DIR • VDE–268
  $FAC • VDE–267
  $FACMODTYP • VDE–267
  $FACROOT • VDE–268
  $FILEDIR • VDE–267
  $FILESPEC • VDE–267
  $LIBROOT • VDE–268
  $MOD • VDE–268
  $MODTYP • VDE–269
  $SCRATCH • VDE–269
  $SESSIONNAME • VDE–269
  $SESSIONNUM • VDE–269
  $STAGEDEST • VDE–269
  $STAGEDIR • VDE–269
  $STAGEROOT • VDE–270
  $STREAM • VDE–270
  $STRMDIR • VDE–270
  $STRMROOT • VDE–270
  $TYP • VDE–270
Script Functions • VDE–267
Script Keyword
  %CURRENT • VDE–80, VDE–266, VDE–268, VDE–270, VDE–353
  %PARENT • VDE–80, VDE–266, VDE–268, VDE–270, VDE–353
Script Types • VDE–271
SCT • VDE–56, VDE–182, VDE–230
SCT notes conference • A–2
SCT Template Files • A–3
SELECT Review Build Job Command • VDE–251
SELECT Review Replacement Command • VDE–255
Session • VDE–269
$SESSIONNAME Script Function • VDE–269
$SESSIONNUM Script Function • VDE–269
SET ARCHITECTURE command • VDE–274
SET CONTEXT command • VDE–275
SET DEVELOPER Command • VDE–277
SET DIRECTORY command • VDE–278
SET EDITOR command • VDE–284
SET FACILITY command • VDE–287
SET KEY command • VDE–288
SET LIBRARY command • VDE–289
SET LOG command • VDE–291
SET NOVERIFY command • VDE–292
SET PRIVILEGES command • VDE–293
SET PROMPT command • VDE–296
SET STREAM command • VDE–297
SET TERMINAL command • VDE–298
SET VERIFY command • VDE–299
Shareable Image
  VDE$RECDEP • B–7
SHOW ARCHITECTURE command • VDE–300
SHOW BUILD_JOB command • VDE–301
SHOW CONTEXT command • VDE–306
SHOW DEFAULTS command • VDE–308
SHOW DEVELOPER Command • VDE–309
SHOW DIRECTORY command • VDE–310
SHOW EDITOR command • VDE–314
SHOW FACILITY command • VDE–316
SHOW FOLD command • VDE–318
SHOW GENERATION command • VDE–321
SHOW GROUP command • VDE–326
SHOW HISTORY command • VDE–321
SHOW KEY command • VDE–329
SHOW LIBRARY command • VDE–289, VDE–331
SHOW LOCATION command • VDE–332
SHOW LOG command • VDE–334
SHOW MEMORY command • VDE–335
SHOW MODULE command • VDE–337
SHOW PRIVILEGES command • VDE–340
SHOW REPLACEMENT command • VDE–342
SHOW RESERVATION command • VDE–346
SHOW Review Replacement Command • VDE–256
SHOW SCRIPT command • VDE–349
SHOW SESSION command • VDE–354
SHOW STEP command • VDE–357
SHOW STREAM command • VDE–360
SHOW SYMBOL command • VDE–364
SHOW TERMINAL command • VDE–365

SHOW USER command • VDE–366
SHOW VERIFY command • VDE–368
SHOW VERSION command • VDE–369
Snapshot Directory • VDE–54
Source Control Tracking • VDE–56, VDE–182
Source Module
  Defining Module As • VDE–67
  Definition • VDE–65
  Display Of • VDE–322, VDE–337
SPAWN command • VDE–370
SPAWN Review Build Job Command • VDE–251
SPAWN Review Replacement Command • VDE–257
$STAGEDEST Script Function • VDE–269
$STAGEDIR Script Function • VDE–269
$STAGEROOT Script Function • VDE–270
Staging Area • VDE–3, VDE–15, VDE–39, VDE–79, VDE–224, VDE–229, VDE–234, VDE–237, VDE–256, VDE–270, VDE–278, VDE–280, VDE–310, VDE–385
START BUILD_JOB command • VDE–373
Statistics Collection • VDE–61, VDE–186, VDE–232, VDE–235
STOP BUILD_JOB command • VDE–376
STOP SUBPROCESS command • VDE–378
$STREAM Script Function • VDE–270
$STRMDIR Script Function • VDE–270
$STRMROOT Script Function • VDE–270
SUSPEND BUILD_JOB command • VDE–379
Symbol
  VDE$WHEREIS_BACKUPS_AFTER_IMAGE • DCL–11, A–3
  VDE$WHEREIS_BACKUPS_ARCHIVE • DCL–11, A–3
  VDE$WHEREIS_BACKUPS_CURRENT • DCL–11, A–3
  VDE$WHEREIS_BACKUP_FULL • DCL–12, A–3
  VDE$WHEREIS_BACKUP_INCR • DCL–11, A–3
  VDE$WHEREIS_BUILD • DCL–12, A–3
  VDE$WHEREIS_CMS • DCL–11, A–3
  VDE$WHEREIS_DATABASE • DCL–11, A–3
  VDE$WHEREIS_LABEL • DCL–11, A–3
  VDE$WHEREIS_MNEMONIC • DCL–11, A–3
  VDE$WHEREIS_STAGING • DCL–12, A–4
  VDE$WHEREIS_STATISTICS • DCL–12, A–4
SYS$LOGIN Logical Name • VDE–138
SYS$OUTPUT • VDE–370
SYS$SCRATCH Logical Name • VDE–138, A–3
SYS$SHARE:VDE$RECDEP.EXE • B–7
System Authorization Database • DCL–9, VDE–134
System Authorization File • DCL–9
SYSUAF • DCL–9, VDE–56, VDE–134, VDE–182

T

$TYP Script Function • VDE–270

U

UIC • VDE–38
UNRESERVE command • VDE–22, VDE–66, VDE–381
Uploading CMS Libraries • D–1

V

VDE$BUILD Utility • VDE–9, VDE–12, VDE–76, VDE–95, VDE–165, VDE–213
VDE$CMS • VDE–40, VDE–62, VDE–281
VDE$DB_AREA • VDE–54
VDE$K_DEPEND_BEGIN_RECORD • B–2
VDE$K_DEPEND_END_RECORD • B–2
VDE$K_DEPEND_INPUT_FILE • B–2
VDE$K_DEPEND_INPUT_LIBRARY • B–2
VDE$K_DEPEND_INPUT_MODULE • B–2
VDE$K_DEPEND_OUTPUT_FILE • B–2
VDE$K_DEPEND_OUTPUT_LIBRARY • B–2
VDE$K_DEPEND_OUTPUT_MODULE • B–2
VDE$K_DEPEND_VERSION • B–2
VDE$LIBRARY • DCL–3, DCL–4, DCL–5, DCL–9, VDE–54, VDE–55, VDE–62, VDE–289
VDE$LIBRARY Logical Name • A–1
VDE$LIBRARY_DEFAULTS logical name • VDE–290
VDE$LIBRARY_DEFAULTS Logical Name • A–1
VDE$LIBRARY_DEFAULTS_CMS Logical Name • A–1
VDE$LIBRARY_DEFAULTS_LABELS logical name • VDE–184, VDE–290
VDE$LIBRARY_DEFAULTS_LABELS Logical Name • A–1
VDE$LIBRARY_DEFAULTS_MNEMONICS logical name • VDE–184, VDE–290
VDE$LIBRARY_DEFAULTS_MNEMONICS Logical Name • A–1
VDE$LIBRARY_LABEL logical name • VDE–184

VDE$LIBRARY_LABEL Logical Name • A–2
VDE$LIBRARY_MNEMONIC logical name • VDE–184
VDE$LIBRARY_MNEMONIC Logical Name • A–2
VDE$MARKER • VDE–69, VDE–281
VDE$NOTES_LIBRARY logical name • VDE–90, VDE–210
VDE$NOTES_LIBRARY Logical Name • A–2
VDE$OFFLINE Logical Name • A–2
VDE$PAGEFILE_SECTION • B–7
VDE$RECDEP • B–7
VDE$RECORD_DEPENDENCY • B–7
VDE$REPLACE_DEFAULT Logical Name • A–2
VDE$SCRATCH Logical Name • VDE–138, A–3
VDE$SNAPSHOT • VDE–54
VDE$TEMPLATES Logical Name • A–3
VDE$UPLOAD • D–1
VDE$V_CI_FLAG_SIGNAL • C–4, C–7, C–9, C–12, C–14, C–16, C–18, C–21, C–23, C–26, C–29
VDE$WHEREIS_BACKUPS_AFTER_IMAGE Logical Name • DCL–11, A–3
VDE$WHEREIS_BACKUPS_AFTER_IMAGESymbol • DCL–11
VDE$WHEREIS_BACKUPS_AFTER_IMAGE Symbol • A–3
VDE$WHEREIS_BACKUPS_ARCHIVE Logical Name • DCL–11, A–3
VDE$WHEREIS_BACKUPS_ARCHIVE Symbol • DCL–11, A–3
VDE$WHEREIS_BACKUPS_CURRENT Logical Name • DCL–11, A–3
VDE$WHEREIS_BACKUPS_CURRENT Symbol • DCL–11, A–3
VDE$WHEREIS_BACKUP_FULL Logical Name • DCL–12, A–3
VDE$WHEREIS_BACKUP_FULL Symbol • DCL–12, A–3
VDE$WHEREIS_BACKUP_INCR Logical Name • DCL–11, A–3
VDE$WHEREIS_BACKUP_INCR Symbol • DCL–11, A–3
VDE$WHEREIS_BUILD Logical Name • DCL–12, A–3
VDE$WHEREIS_BUILD Symbol • DCL–12, A–3
VDE$WHEREIS_CMS Logical Name • DCL–11, A–3
VDE$WHEREIS_CMS Symbol • DCL–11, A–3
VDE$WHEREIS_DATABASE Logical Name • DCL–11, A–3
VDE$WHEREIS_DATABASE Symbol • DCL–11, A–3
VDE$WHEREIS_LABEL Logical Name • DCL–11, A–3
VDE$WHEREIS_LABEL Symbol • DCL–11, A–3
VDE$WHEREIS_MNEMONIC Logical Name • DCL–11, A–3
VDE$WHEREIS_MNEMONIC Symbol • DCL–11, A–3
VDE$WHEREIS_STAGING Logical Name • DCL–12, A–4
VDE$WHEREIS_STAGING Symbol • DCL–12, A–4
VDE$WHEREIS_STATISTICS Logical Name • DCL–12, A–4
VDE$WHEREIS_STATISTICS Symbol • DCL–12, A–4
VDE/DEPEND command • DCL–6
VDE/READ_UAF command • DCL–9, VDE–134
VDE/WHEREIS command • DCL–11
VDE command
  at DCL level • DCL–2
VDE Utility
  Command Abbreviations • 1–3
  commands • VDE–2 to VDE–389
  exiting from • 1–2, DCL–3
  HELP command • 1–3
  invoking • 1–2
VDE vs VSC • 1–2
VERIFY GENERATION command • VDE–384
VSC Library • VDE–289
VSC Utility
  HELP command • 1–3
  invoking • 1–2
VSC vs VDE • 1–2

W

WAIT BUILD_JOB command • VDE–388
Wildcard characters
  deleting facilities with • VDE–114
  deleting groups using • VDE–120
  deleting modules with • VDE–123
  deleting streams with • VDE–132
  deleting usernames with • VDE–134