VMS SES
Security Manager's Guide
Version 5.2

Order Number: QS-970-MG

November 1989

This manual describes the underlying concepts and user accessible characteristics of the Version 5.2 VMS Security Enhancement Service. It is primarily intended to assist a security manager in the configuration and use of a VMS Security Enhancement Service mandatory control environment.

Revision/Update Information:	This manual supersedes the Version 5.1 *VMS SES Security Manager's Guide*.
Operating System and Version:	VMS Version 5.2
Software Version:	SEVMS Version 5.2

Digital Equipment Corporation

November 1989

The postpaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist in preparing future documentation.

This document was prepared using VAX DOCUMENT, Version 1.1

Contents

PREFACE v

CHAPTER 1 INTRODUCTION TO SEVMS 1-1

1.1 INTENT OF SEVMS 1-1

1.2 DESCRIPTION OF SEVMS 1-1

1.3 COMPLEMENTARY SECURITY TECHNIQUES OF VMS AND SEVMS 1-2

CHAPTER 2 OVERVIEW OF SEVMS 2-1

2.1 MANDATORY ACCESS CONTROLS 2-1

2.2 THE SECURITY MODEL 2-1
2.2.1 Subjects 2-1
2.2.1.1 Subject Identification • 2-2
2.2.2 Objects 2-2
2.2.3 Secrecy 2-2
2.2.4 Integrity 2-3
2.2.5 Labels 2-3

2.3 CLASSIFICATION STRINGS 2-4
2.3.1 Input Classification Strings 2-4
2.3.1.1 Classification Levels • 2-5
2.3.1.2 Classification Categories • 2-6
2.3.1.3 Classification Ranges • 2-6
2.3.2 Output Strings 2-7

CHAPTER 3 PROTECTION FEATURES 3-1

3.1 DETERMINING ACCESS 3-1

3.1.1 SEVMS Object Support 3-1
3.1.2 Printer Access Protection 3-4
3.1.3 How Privileges Affect Access 3-4
3.1.4 Propagation of Classification 3-4

3.2 FILE PROTECTION FEATURES 3-5
3.2.1 I/O Operations and ODS-2 Access Checks 3-5
3.2.2 Access Accountability 3-7

CHAPTER 4 MANDATORY ACCESS CONTROL AND THE USER 4-1

4.1 LOGGING INTO THE SYSTEM 4-1
4.1.1 Specification of Session Classification 4-1
4.1.1.1 Captive Accounts • 4-2
4.1.1.2 Batch Processes • 4-2
4.1.1.3 Network Processes • 4-2
4.1.1.4 Detached Processes • 4-2

4.2 DISPLAYING CLASSIFICATION OF A PROCESS 4-2

4.3 CLASSIFICATION 4-3
4.3.1 Determining Object Classification 4-3
4.3.2 Organizing Classified Directories 4-4
4.3.2.1 Protecting Directory Names • 4-4
4.3.2.2 Directory Creation by Privileged Users • 4-5
4.3.2.3 Directory Creation by Unprivileged Users • 4-5
4.3.3 Name Spaces 4-5

4.4 PRINTING FILES 4-6

CHAPTER 5 MANDATORY ACCESS CONTROL AND THE SECURITY MANAGER 5-1

5.1 SECURITY MANAGEMENT 5-1

5.2 PRIVILEGES 5-2

5.3 DEFINING SECRECY LEVELS 5-2

5.4 SECRECY CATEGORIES 5-3
5.4.1 Description 5-4
5.4.2 Defining Secrecy Categories 5-4

5.5 CHANGING CLASSIFICATION 5-6
5.5.1 UPGRADE Privilege 5-6

5.6 ENABLING AND DISABLING MANDATORY CONTROLS 5-6
5.6.1 DOWNGRADE Privilege 5-6
5.6.2 BYPASS Privilege 5-7

5.7 PROTECTING CRITICAL OBJECTS 5-7
5.7.1 Classifying Devices 5-8
5.7.1.1 Terminals • 5-8
5.7.1.2 VAXstations • 5-8
5.7.1.2.1 VWS Window Manager Support • 5-8
5.7.1.2.2 DECwindows • 5-9
5.7.1.3 Disk Devices • 5-9
5.7.1.4 Tape Devices • 5-9
5.7.2 Classifying Volumes 5-9
5.7.2.1 Protecting Files • 5-10
5.7.2.1.1 Protecting SYS$SYSTEM • 5-11
5.7.2.2 Protecting Directories • 5-11
5.7.3 Classifying Global Sections 5-11
5.7.4 Classifying Logical Name Tables 5-12
5.7.5 Classifying Queues 5-12
5.7.6 Classifying Mailboxes 5-13

5.8 CLASSIFICATION AND INTERACTIVE LOG-INS 5-13
5.8.1 Classifying Accounts 5-14
5.8.1.1 Setting Classifications for the SYSTEM Account • 5-16
5.8.2 Classifying Terminals Connected by Serial Line Interfaces 5-16
5.8.3 Classifying Terminals Connected by DECnet 5-16
5.8.4 Classifying Terminal Server Ports 5-16
5.8.4.1 Interactive Ports • 5-17
5.8.4.2 Application and Dedicated Ports • 5-18
5.8.5 Privileges and Log-in Classification 5-18

5.9 NETWORKS IN THE CLASSIFIED ENVIRONMENT 5-19
5.9.1 Session Control 5-19
5.9.2 DECnet Connections Between SEVMS and Non-SEVMS Nodes 5-20

5.9.3 DECnet Proxy Access — 5-21
5.9.4 Restricting Network Access — 5-21
5.9.5 Network File Transfers — 5-22
5.9.6 Remote Terminal Sessions — 5-22
5.9.6.1 Classifying RTA0: • 5-22
5.9.6.2 Associating Remote Link Classifications • 5-23
5.9.7 Associating Classifications With Nodes — 5-23

5.10 BACKUP UTILITY IN SEVMS — 5-24

5.11 MAIL IN THE CLASSIFIED ENVIRONMENT — 5-24

5.12 DECWINDOWS IN THE CLASSIFIED ENVIRONMENT — 5-26

CHAPTER 6 AUDITING — 6-1

6.1 INTRODUCTION — 6-1

6.2 SEVMS AUDITING COMMANDS AND QUALIFIERS — 6-2
6.2.1 SET AUDIT — 6-2
6.2.1.1 SET AUDIT/ALARM • 6-2
6.2.1.2 SET AUDIT/SECRECY • 6-3
6.2.2 SET AUDIT/INTEGRITY — 6-4
6.2.3 SHOW AUDIT — 6-4
6.2.4 ANALYZE/AUDIT — 6-5

6.3 ALARM MESSAGES — 6-6
6.3.1 Print Symbiont Alarms — 6-6
6.3.2 File Access Alarms — 6-7
6.3.3 Classification Change Alarms — 6-8
6.3.4 Binary Audit Record Format — 6-9

CHAPTER 7 THE SEVMS SECURE PRINT FACILITY — 7-1

7.1 INTRODUCTION — 7-1

7.2 PRINT SYMBIONT LOGICAL NAMES — 7-1

7.3 PRINT SYMBIONT TEMPLATES 7-2
7.3.1 Selecting a Template 7-3
7.3.1.1 Overview • 7-3
7.3.1.2 Template Selection Process • 7-3
7.3.2 Establishing and Using Templates 7-4
7.3.2.1 Step 1 - Determining Templates Needed • 7-5
7.3.2.1.1 Front and Back Page Considerations • 7-5
7.3.2.1.2 Graphics Printer Considerations • 7-6
7.3.2.2 Step 2 - Designing Templates • 7-7
7.3.2.2.1 Template Elements • 7-8
7.3.2.2.2 Element Directives • 7-8
7.3.2.2.3 Field Directives • 7-9
7.3.2.2.4 Format Directives • 7-12
7.3.2.2.5 Directive Usage Notes • 7-13
7.3.2.3 Step 3 - Storing Templates • 7-13
7.3.2.4 Step 4 - Associating Templates • 7-13
7.3.2.4.1 Template Association • 7-14
7.3.2.4.2 SEVMS Logical Printers • 7-14
7.3.2.4.3 Examples • 7-15
7.3.2.4.4 Using the SET and SHOW TEMPLATE Commands • 7-16

7.4 CUSTOMER-SUPPLIED FORMATTING ROUTINES 7-16
7.4.1 Introduction 7-16
7.4.2 The Symbiont/Formatting-Routine Interface 7-17
7.4.3 Creating and Using Formatting Routines 7-18
7.4.4 Usage Notes 7-18

7.5 INVOKING A PRINT SYMBIONT 7-19

7.6 PRINTING LISTING FILES 7-20

7.7 USING PRINT SYMBIONT RESET ROUTINES 7-20

APPENDIX A SEVMS_DEFAULT SYMBIONT TEMPLATE A-1

INDEX

TABLES

3-1	SEVMS Object Support	3-2
3-2	Mandatory Access Control Algorithm	3-3
3-3	SEVMS Printer Access Protection	3-4
6-1	New Keywords For SET AUDIT/ALARM	6-2
7-1	Lengths of Field Directives	7-9
7-2	Sample template associations	7-14

Preface

The *VMS SES Security Manager's Guide* describes the underlying concepts and user configurable characteristics of the Version 5.2 VMS Security Enhancement Service software.

The VMS Security Enhancement Service (VMS SES) is a software security consulting package. It provides many features of mandatory access controls and security auditing for the VMS operating system.

The VMS SES software security consulting package is composed of the following components:

- Services performed by a DIGITAL consultant
- Licensed software
- Documentation

VMS SES provides the services of a trained DIGITAL consultant who supports the customer in several areas, such as: assisting in planning security policies and controls, training users, and installing the licensed software.

The licensed software component of this product is called SEVMS. SEVMS provides a tool set for devising a system-wide security policy to help safeguard users, data, and software from security threats. Since this manual describes the features of the licensed software, the term SEVMS is used throughout this manual to reference this software. SEVMS is also the VMS facility name for the licensed software and is used as a prefix for many of the software components.

A documentation set which describes the SEVMS software and how it is installed, used, and managed is provided with the VMS SES package.

Intended Audience

This document makes the general assumption that the reader is a system manager or security manager. It is intended to assist in the system configuration of the VMS Security Enhancement Service (VMS SES) mandatory control environment.

The security manager in a given environment may also be, in some cases, the system manager. If the security manager and system manager are different individuals, both should review this material and mutually implement the secure environment.

Since this manual describes the underlying concepts of SEVMS, it may also be of interest to system designers and programmers.

Document Structure

The information in this document is divided into the following topics:

- Introduction
- Overview
- Protection Features
- Mandatory Access Control and the User
- Mandatory Access Control and the Security Manager
- Auditing
- SEVMS Print Symbiont
- SEVMS Default Symbiont Template
- SEVMS System Messages and Recovery Procedures
- SEVMS Binary Audit Record Format

The introductory sections of this document, Chapters 1 through 5, explain the underlying concepts and user-configurable characteristics of SEVMS and are of interest to system designers and programmers, as well as security managers.

The remaining sections, Chapters 6 and 7, involve issues of concern only to the security manager. These chapters explicitly address the installation and configuration of the SEVMS mandatory control environment.

The appendices at the end of the manual include supplemental information.

Associated Documentation

This manual should be used in conjunction with the other manuals of the SES document set and the manuals of the VMS document set. References will be made throughout this manual to VMS SES manuals and VMS manuals.

SES Document Set

This manual is one of three manuals that form the VMS Security Enhancement Service (SES) document set. This document set consists of the following manuals:

- *VMS SES Installation Guide and Release Notes* — This manual is intended as a supplemental manual of the SES documentation set. It provides information concerning the installation (but not configuration) of SEVMS on a VMS system. It also contains release notes which summarize omitted features, resolved problems, new features, and known problems and restrictions for the current release of the SEVMS software.

- *VMS SES User's Guide* — This manual describes the mandatory protection mechanisms provided by SEVMS, the interaction of these mechanisms with VMS discretionary protection mechanisms, and the use of commands and utilities which are unique to SEVMS. It is intended for all SEVMS users.
- *VMS SES Security Manager's Guide* — This manual describes the configuration, management, and operation of SEVMS. It is intended for use by system administrators and security officers. This manual assumes that the reader is familiar with basic VMS security practices and the VMS documentation which describes VMS security.

Together, these manuals form complete documentation about SEVMS. For information about related VMS features and functions, the user should refer to the manuals of the VMS document set.

VMS Documentation Set

The VMS documentation set has two main divisions:

- VMS Base Documentation Set
- VMS Extended Documentation Set

The VMS Base Documentation Set is a desk-top set for users of small standalone systems and low-end Local Area VAXclusters, and for general users of large VAX systems. The Base Documentation Set contains concise, easy to find, information about performing day-to-day tasks. This documentation set contains the following components:

- Overview of VMS Documentation
- VMS New Features Manual
- VMS General User's Manual
- VMS System Manager's Manual
- VMS Mini-Reference Manual
- VMS License Management Manual

The VMS Extended Documentation Set is a full documentation set for users who need more detail about any VMS component to perform daily tasks. The Extended Documentation Set also meets the needs of system managers of large VAX systems and of system and application programmers.

This documentation set contains the following components:

- General User Subkit
- System Management Subkit
- Programming Subkit

These manuals are supplemented by several other forms of VMS documentation: Release Notes, Obsolete Features Kit, Software Installation and Operations Guides, online help information, and other optional documentation.

Refer to the *Overview of VMS Documentation* booklet in the VMS documentation set for complete information about the VMS documentation set.

Relationship Between VMS and SEVMS Documentation

The documentation for SEVMS is intended to be used along with the documentation for VMS. While the SES documentation set addresses issues specific to the SEVMS product, issues of a more general nature pertaining to VMS are addressed in the VMS documentation set. Therefore, you should consider the manuals of the SES document set to be an extension of your existing VMS document set. As such, SES manuals do not repeat information already contained in existing VMS documentation. Instead, references are made throughout SES manuals to several of the manuals in the VMS document set when appropriate. The following VMS documentation is most frequently referenced by the SES manuals:

- *VMS System Management Subkit*
- *Guide to VMS System Security*
- *VMS Audit Analysis Utility Manual*
- *VMS DCL Dictionary*
- *VMS Release Notes*
- *VMS Audit Analysis Utility Manual*

It is assumed that the reader is especially familiar with the contents of the *Guide to VMS System Security* and the *VMS System Management Subkit*.

Conventions

This section describes the VMS and SEVMS conventions which are used in this manual.

VMS Conventions Used in This Manual

Throughout this manual, the following standard conventions are used in examples of commands:

Convention	Meaning
[ ]	Square brackets indicate that the enclosed item is optional.
{ }	Braces enclose a list from which one element must be selected.
< >	Angle brackets indicate that item is to be replaced by a specific instance of the named quantity.
|	The OR symbol separates alternatives within braces or brackets.
. . .	An ellipsis indicates that the preceding item(s) can be repeated one or more times.
:=	A "colon equals" indicates the item to its left is defined as the item to its right.

Unless otherwise indicated in the examples, commands are terminated by pressing the Return key.

Colons (:) and equals signs (=) are used interchangeably in descriptions of DCL command qualifiers.

SEVMS Conventions Used in This Manual

SEVMS mandatory access controls introduce a number of new protection attributes and relationships. Among these are the concepts of hierarchical numeric *levels* and non-numeric non-hierarchical *categories*. Categories form discrete mathematical *sets*.

The *operators* used to indicate the relationship between numeric quantities *(scalar)* differ from the operators used to indicate the relationship between non-numeric quantities *(sets)*, although their meanings are similar. The operators used in this manual are described and compared in the following table.

Operator	Scalar interpretation for Security Levels	Operator	Set interpretation for Security Categories
$<$	is less than	$\subset$	is a proper subset of
$\leq$	is less than or equal	$\subseteq$	is a subset of
$=$	is equal to	$\equiv$	is identical to
$>$	is greater than	$\supset$	is a proper superset of
$\geq$	is greater than or equal	$\supseteq$	is a superset of
$\neq$	is not equal to	$\not\equiv$	is not identical to

In informal discussions of the relationship between two classifications, the scalar relationships may be used to refer to both the scalar (level) and set (categories) portions of the classification. For instance, the informal statement "A's classification is equal to B's" means "A's level = B's level AND A's categories $\equiv$ B's categories".

Dominates describes a relationship between two classifications. "A's classification dominates B's" means "A's level $\geq$ B's level AND A's categories $\supseteq$ B's categories".

1 Introduction To SEVMS

This chapter introduces and describes the nature of SEVMS, defines its intent, and discusses its relationship to VMS.

1.1 Intent of SEVMS

It is important to note that the SEVMS mandatory controls do not replace the familiar VMS discretionary access controls (such as *access control lists*). Instead, SEVMS mandatory controls are used in addition to standard VMS discretionary controls, and *augment* VMS protection mechanisms. Therefore, SEVMS is intended to provide the system security manager with a means to enforce an additional system-wide mandatory access control policy.

1.2 Description of SEVMS

SEVMS implements a *non-discretionary,* or *mandatory,* access control mechanism. The most distinguishing characteristic of the mandatory access control mechanism is that the security policy is *beyond direct user control.* This means that security policy is centrally and uniformly established by the system security manager, often the system manager. SEVMS is responsible for the enforcement of the security policy established by the security manager.

The use of SEVMS mandatory controls enables the security manager to classify users and data with different levels of sensitivity. By classifying users, SEVMS ensures the following:

- Users cannot read data unless their classification permits it.
- Users cannot write data with a new classification that would grant read access to users who could not previously read the data.

In addition to classifying users, SEVMS also provides the following capabilities:

- It provides auditing of attempts to compromise mandatory access controls.
- It enforces the segregation of certain users to certain terminals, based on their classification.
- It enforces the segregation of printed output to certain printers, based on their classification.
- It enforces the segregation of files to certain disks, based on their classification.
- It provides a uniform mechanism for the sensitivity labeling of print jobs.

1.3 Complementary Security Techniques of VMS and SEVMS

As mentioned at the beginning of this chapter, SEVMS security features are *not* meant to replace standard VMS security features. Instead, the security features of SEVMS are provided in addition to those standard features of VMS. Before implementing SEVMS security features, it is important that the security manager be very familiar with VMS security features and how they are used. Therefore, the material contained in the *Guide to VMS System Security* should be thoroughly read and understood.

2 Overview of SEVMS

This chapter provides an overview of the basic aspects of the mandatory access controls provided by SEVMS. More detailed information can be found in Chapter 3.

Information about the discretionary access controls provided by VMS is found in the *Guide to VMS System Security*.

2.1 Mandatory Access Controls

SEVMS augments the UIC-based and ACL-based protection of VMS with *mandatory access controls*, also referred to as *non-discretionary access controls* in this manual. Mandatory access controls enable a *classification* to be assigned to an object, for example, a file or device, which cannot be reduced, except by a privileged user. The operating system enforces this protection, preventing users who have access to the classified information from passing that data to an untrusted part of the system.

Because users with certain privileges may circumvent the mandatory security policy (indeed, *any* security policy), these privileges must be restricted to "trusted" personnel. Generally, only the system manager and security manager, if different, retain the use of these privileges in the classified environment.

2.2 The Security Model

SEVMS conforms to the *Reference Monitor* concept of a secure system. This model is explained in detail in the *Guide to VMS System Security*. An overview of the Reference Monitor concept is provided in this section.

2.2.1 Subjects

A *subject* is an active component of the system which utilizes computational elements, known as *objects*, to perform some useful function. Subjects possess a permanent attribute known as a *clearance* [1] , which is utilized to regulate access to objects.

The only subjects on an SEVMS system are processes (i.e. all processes are subjects.). (Users may informally be considered subjects, but they actually perform their actions through processes.)

[1] Internally, SEVMS represents clearances and classifications by identical data structures, and these will be referred to generically as classifications for both subjects and objects.

2.2.1.1 Subject Identification

When a user logs into the system, a clearance (classification), must be associated with the session. This is accomplished through the normal system log-in mechanism operating in conjunction with user clearance information which is present in the system authorization file (SYSUAF.DAT).

2.2.2 Objects

An *object* is a computational element which serves either as a sink of data, a source of data, or both. Objects possess an attribute known as a *classification*, which is used in conjunction with a subject's classification to determine access rights.

The most common objects within VMS are files, devices, and mailboxes. Since a process may be the target of certain system services, a process may, in addition to being a subject, also be an object. The formal relationship between subject and object classifications comprise the central theme of the SEVMS mandatory control scheme.

2.2.3 Secrecy

SEVMS implements a *lattice security model,* [1] but since the term *security* is already used to refer to other VMS features (and used as a command qualifier) SEVMS refers to the properties associated with the model as *secrecy*. The basic principle behind the lattice security model is that no subject can read an object which is more classified than the subject (the simple security property), and that no subject can write to an object less classified than the subject (the *-security property). More information on the SEVMS implementation of the lattice security model can be found in Chapter 3.

Secrecy is utilized to preserve the *confidentiality* of the data contained in a classified object.

The secrecy of an object is defined by the following two factors:

- *Secrecy level* - A level ranging from 0 - 255.
- *Secrecy categories* - Up to 128 categories.

Secrecy levels and secrecy categories are described in the *VMS SES User's Guide*.

[1] As described in Secure Computer Systems: Unified Exposition and Multics Interpretation, Bell, D.E. and LaPadula, L.J., MTR-2997, MITRE Corp., Bedford, MA, March 1976

2.2.4 Integrity

SEVMS also implements a *lattice integrity model.*[2] The basic principle behind the lattice integrity model is that no subject can read an object that has less integrity than the subject, and that no subject can write to an object that has more integrity than the subject. More information on the SEVMS implementation of the lattice integrity model can be found in Chapter 3.

The integrity property is utilized to assure the *trustworthiness* of the data contained in a classified object.

The *integrity* of an object is defined by the following two factors:

- *Integrity level* - A level in the range of 0 - 255.
- *Integrity categories*—Up to 64 categories.

Note: Although SEVMS provides controls for integrity, the consequences of its use with VMS are not well understood. Therefore, DIGITAL supports SEVMS' detailed functionality in regards to integrity controls, but cannot make any assurances as to the overall consequences of using those controls. In particular, *the VMS Security Enhancement Service does not provide assistance in the use of integrity at this time.*

2.2.5 Labels

Each subject and classified object has its classification kept within a *label.* A label consists of the following four parts:

- A secrecy level
- A set of secrecy categories (there may be none)
- An integrity level
- A set of integrity categories (there may be none)

Within the system, a label is kept in a data structure known as a *classification block.* A classification block is 20 bytes long and contains the label in a format that is not human readable. Externally, labels are input and output as human readable *classification strings.* Classification strings are explained in Section 2.3.

Some objects can be assigned a classification range. They are known as *ranged* objects and have two classifications associated with them:

- A minimum classification which defines the low end of the range
- A maximum classification which defines the high end of the range.

More information on ranged objects appears in Chapter 3.

[2] As described in Integrity Considerations for Secure Computer Systems, Biba, K.J., ESD-TR-76-372, Electronic System Division, AFSC, Hanscom AFB, MA, April 1977

2.3 Classification Strings

SEVMS provides a standard human readable format for expressing classification labels that is used whenever a label must be input into the system (for example, as in a DCL command) or displayed by the system (for example, as in a listing). While the system stores the secrecy and integrity portions of a classification in the same internal label, externally they are specified and displayed separately. In fact, because integrity is infrequently used (see the note at the end of Section 2.2.4), utilities may not display the integrity if the integrity level is zero and there are no integrity categories.

This section discusses two types of classification strings, *input strings* and *output strings*.

2.3.1 Input Classification Strings

Input classification strings are used to provide a command or utility with classification information.

The format of an input classification string is one of the following:

- (LEVEL:<level or range>)
- (CATEGORY:(<categories or range>))
- (LEVEL:<level or range>, CATEGORY:(<categories or range>))

The secrecy (or integrity) level and categories are NOT separable; if only one is specified, the other is implied.

If a level is not specified, zero is assumed. If categories are not specified, an empty set is assumed.

A range, rather than a single level or set of categories, can be specified if the object being referred to is a ranged object. More information on the format for ranged objects is located in Section 2.3.1.3.

Some examples of classification strings, as used with a secrecy qualifier, are shown below:

Input strings

1
```
/SECRECY=(LEVEL=UNCLASSIFIED)
```

This indicates a secrecy level of UNCLASSIFIED and no secrecy categories. If categories are not specified, the default is none. Integrity is unaffected.

2
```
/SECRECY=(LEVEL=9,CATEGORY=(1,2,3))
```

This indicates a secrecy level of 9 and secrecy categories of 1, 2, and 3. Integrity is unaffected.

3
```
/SECRECY=(LEVEL=SECRET)/INTEGRITY=(LEVEL:1, CATEGORY:(BLUE,GREEN))
```

This indicates a secrecy level of SECRET, an integrity level of 1, no secrecy categories, and integrity categories BLUE and GREEN.

4
```
/SECRECY=(CATEGORY=RED)
```

This indicates a secrecy level of 0 and a secrecy category of RED. Integrity is unaffected. If a level is not specified, the default is 0.

2.3.1.1 Classification Levels

Classification levels assign hierarchical classification information to an object. Two different levels can be assigned to an object: a *secrecy level* and an *integrity level.*

In each case, a level is a numeric value which ranges from 0 to 255. In practice, however, levels are usually referred to using a textual identifier defined in the rights database.

Note: **For simplicity's sake, the examples in this section deal only with levels, not categories. In actual operation, any desired categories must also be specified as part of the classification string. In these examples, the categories have the default value of "none".**

When it is necessary to specify a secrecy and/or an integrity level as a qualifier to a command, the following syntax is used:

```
/SECRECY = (LEVEL = <level>)
/INTEGRITY = (LEVEL = <level>)
```

For example, if the secrecy level of TOP_SECRET (numeric level 255) and the integrity level of GOOD_STUFF (numeric level 100) have been defined in the rights database by the system manager, the following specifications are correct.

```
/SECRECY = (LEVEL = TOP_SECRET)
/INTEGRITY = (LEVEL = GOOD_STUFF)
```

```
/SECRECY = (LEVEL:255)
/INTEGRITY = (LEVEL:100)
```

Secrecy and integrity levels are *distinct;* you may not mix them. The following specifications are *not* correct, given the definitions of TOP_SECRET and GOOD_STUFF, and generate an error if attempted.

```
/SECRECY = (LEVEL = GOOD_STUFF)
/INTEGRITY = (LEVEL = TOP_SECRET)
```

Identifiers for levels are generally defined in the rights database, however, a numeric value can be used for a level whether or not a textual identifier has been defined for it.

2.3.1.2 Classification Categories

Classification categories assign non-hierarchical classification information to an object. Two different sets of categories can be assigned to an object: *secrecy categories* and *integrity categories*.

SEVMS provides up to 128 secrecy categories and up to 64 integrity categories. The sets of categories are separate, just as secrecy and integrity levels are separate.

In each case, a category is a numeric value. In practice, however, categories are often referred to using a textual identifier in the rights database.

Note: **For simplicity's sake, the examples in this section deal only with categories, not with levels. In actual operation, the desired level must also be specified as part of the classification string. In these examples, the levels are a default of "0".**

When it is necessary to specify secrecy and integrity categories, the following syntax is used:

```
/SECRECY = (CATEGORY:(<category>[,<category> . . . ]))
/INTEGRITY = (CATEGORY:(<category>[,<category> . . . ]))
```

For example, if the secrecy categories of ALPHA, BETA and GAMMA (numeric values 10, 20, and 30 respectively), and the integrity categories of GOOD, BETTER and BEST (numeric values 1, 2, and 3 respectively) have been defined in the rights database by the system manager, the following specifications are legal:

```
/SECRECY = (CATEGORY:(ALPHA,BETA,GAMMA))
/INTEGRITY = (CATEGORY:(GOOD,BETTER,BEST))

/SECRECY = (CATEGORY:(10,20,30))
/INTEGRITY = (CATEGORY:(1,2,3))
```

Secrecy and integrity categories are distinct; you may not mix them. The following specifications are *not* legal, given the preceding definitions, and generate an error if attempted.

```
/SECRECY = (CATEGORY:(GOOD,BETTER,BEST))
/INTEGRITY = (CATEGORY:(ALPHA,BETA,GAMMA))
```

2.3.1.3 Classification Ranges

Ranged objects have both *minimum* and *maximum* classifications. The minimum and maximum classification for an object defines its classification range; this range includes its endpoints. If the minimum and maximum classifications are equal, the syntax described in Section 2.3.1.1 and Section 2.3.1.2 can be used. There are separate ranges for secrecy and integrity.

The following syntax is used when specifying a level or category range:

```
LEVEL = (MAXIMUM:<level>)
LEVEL = (MINIMUM:<level>, MAXIMUM:<level>)

CATEGORY = (MAXIMUM:<category set>)
CATEGORY = (MINIMUM:<category set>, MAXIMUM:<category set>)
```

The default for an unspecified minimum or maximum level is "0". The default for an unspecified minimum or maximum category set is "none".

For a classification range to be valid, the following relationships between the minimum and maximum classifications must be true:

- The minimum secrecy level must be less than, or equal to, the maximum secrecy level.
- The minimum secrecy categories (if any) must be a subset of, or the same as, the maximum secrecy categories.
- The minimum integrity level must be less than, or equal to, the maximum integrity level.
- The minimum integrity categories (if any) must be a subset of, or the same as, the maximum integrity categories.

2.3.2 Output Strings

Output classification strings are provided in listings and displays to indicate classification information. The format of an output classification string is one of the following:

- SECRECY = (LEVEL:<level or range>)
- SECRECY = (LEVEL:<level or range>, CATEGORY:(<categories or range))
- INTEGRITY = (LEVEL:<level or range>)
- INTEGRITY = (LEVEL:<level or range>, CATEGORY:(<categories or range>))

When the context is clear, in actual output the "SECRECY=" and "INTEGRITY=" prefixes may not be displayed.

The following are examples of output strings.

Examples

1
```
SECRECY=(LEVEL=UNCLASSIFIED)
```

This indicates a secrecy level of UNCLASSIFIED, an integrity level of 0 and no integrity or secrecy categories.

2
```
SECRECY=(LEVEL=9,CATEGORY=(1,2,3))
```

This indicates a secrecy level of 9, secrecy categories of 1, 2, and 3, an integrity level of 0 and no integrity categories.

3
```
SECRECY=(LEVEL=SECRET)
INTEGRITY=(LEVEL:1, CATEGORY:(BLUE,GREEN))
```

This indicates a secrecy level of SECRET, an integrity level of 1, no secrecy categories and integrity categories BLUE and GREEN.

4 `SECRECY=(LEVEL=(MINIMUM=UNCLASSIFIED, MAXIMUM=SECRET), CATEGORY=(NONE))`

This indicates a secrecy level range whose minimum is unclassified and whose maximum is secret; there are no secrecy categories.

3 Protection Features

This chapter describes the protection features of SEVMS. It contains the following topics:

- How an SEVMS system determines access
- File protection features of SEVMS

3.1 Determining Access

This section describes how the system determines access. It contains information about the following topics:

- SEVMS object support
- Printer access protection
- How privileges affect access
- Propagation of classification

Under SEVMS, subjects and objects have protection structures associated with them. For a subject, the structure is called an *Access Rights Block*, or ARB. For an object, the structure is called an *Object's Rights Block*, or ORB.

Both the ARB and ORB data structures contain the following information:

- Secrecy level
- Secrecy categories
- Integrity level
- Integrity category
- Discretionary access control information.

3.1.1 SEVMS Object Support

The objects of VMS, and the level of support offered by SEVMS for each object, are listed in Table 3-1, which follows.

Table 3-1 SEVMS Object Support

Object	SEVMS Object	Ranged	Set By
Disk Volumes	Yes	Yes	$ INITIALIZE
Disk Drives	Yes	Yes	$ SET CLASS/OBJECT = DEVICE
Event Flag Clusters	No	N/A	N/A
Files	Yes	No	$ SET CLASS[/OBJECT = FILE][1]
Group Global Sections	Yes	No[3]	$ SET CLASS/OBJECT = GROUP_GLOBAL_SECTION[2]
Printers	Yes	Yes	$ SET CLASS/OBJECT = DEVICE
Logical Name Tables	Yes	Yes	$ SET CLASS/OBJECT = LOGICAL_NAME_TABLE[1]
Mailboxes	Yes	Yes	$ SET CLASS/OBJECT = DEVICE[1]
Magnetic Tape Drives	Yes	Yes	$ SET CLASS/OBJECT = DEVICE
Process	Yes	No	LOGINOUT
Queue	Yes	Yes	$ SET CLASS/OBJECT = QUEUE
System Global Sections	Yes	No[3]	$ SET CLASS/OBJECT = SYSTEM_GLOBAL_SECTION[2]
Terminals	Yes	Yes	$ SET CLASS/OBJECT = DEVICE

[1]The object is given the creating process' classification when created. This command can be used to change it.

[2]If global section is backed by a data file, it is given the file's classification when created, and the classification cannot be changed. Otherwise, it is given the creating process' classification. This command can be used to change it.

[3]A section not backed by a data file (PFN, Demand Zero, etc.) can have a classification range and is given the creating process' classification when created. This command can be used to change it.

When a subject attempts to access an object which is supported by SEVMS, the following access checking occurs:

- Access mode check— The access mode (kernel, executive, supervisor or user) of the request is compared to that of the object. The request is immediately rejected if a less privileged mode is attempting to access an object protected at a more privileged mode.

- Mandatory access check—This consists of the following checks:
  - For read access:
    - The subject's secrecy classification must dominate the object's minimum secrecy classification.
    - The object's maximum integrity classification must dominate the subject's integrity classification.
  - For write access:
    - The object's maximum secrecy classification must dominate the subject's secrecy classification.
    - The subject's integrity classification must dominate the object's minimum integrity classification.
- Discretionary access checks— The normal VMS access checking procedures are followed; that is, UIC protection masks and Access Control Lists are checked. This topic is fully documented in the *Guide to VMS System Security*.

The mandatory access tests are summarized in Table 3-2, which follows.

Table 3-2 Mandatory Access Control Algorithm

Access Type	Secrecy Property	Integrity Property
READ	SL(subject) $\geq$ SLmin(object) AND SC(subject) $\supseteq$ SCmin(object)	IL(subject) $\leq$ ILmax(object) AND IC(subject) $\subseteq$ ICmax(object)
WRITE	SL(subject) $\leq$ SLmax(object) AND SC(subject) $\subseteq$ SCmax(object) OR subject has DOWNGRADE privilege	IL(subject) $\geq$ ILmin(object) AND IC(subject) $\supseteq$ ICmin(object) OR subject has UPGRADE privilege

SL = Secrecy Level IL = Integrity Level
SC = Secrecy Categories IC = Integrity Categories

Note: Although SEVMS provides controls for integrity, the consequences of its use with VMS are not well understood. Therefore, DIGITAL supports the detailed functionality of SEVMS in regards to integrity controls, but cannot make any assurances as to the overall consequences of using those controls. In particular, *VMS Security Enhancement Service does not provide assistance in the use of integrity at this time.*

3.1.2 Printer Access Protection

In SEVMS, the print symbiont performs additional protection checks at the time the file is printed to assure that the file's classification falls within the classification range of the target printer. If the classification of a file that is submitted for printing is less than or greater than the classification range of the target printer, an access violation occurs and the file will not be printed.

The following table, Table 3-3, summarizes the various conditions of SEVMS printer access protection.

Table 3-3 SEVMS Printer Access Protection

Classification of File and Printer	Result
FILE_CLASS is greater than or equal to PRINTER_CLASS_MINIMUM	
and	File is successfully printed.
FILE_CLASS is less than or equal to PRINTER_CLASS_MAXIMUM	
FILE_CLASS is less than PRINTER_CLASS_MINIMUM	File is not printed. Access Violation.
FILE_CLASS is greater than PRINTER_CLASS_MAXIMUM	File is not printed. Access Violation.

3.1.3 How Privileges Affect Access

If the mandatory access check fails, the desired access to the object may still be granted if the subject possesses one of the following privileges:

- BYPASS - Unconditionally grants full access.
- READALL - Unconditionally grants read access.
- DOWNGRADE - Allows write access to a lower secrecy object.
- UPGRADE - Allows write access to a higher integrity object.
- VOLPRO - Bypasses volume protection checks.
- SECURITY - Allows classification of a ranged object to unequal minimum and maximum classifications.

3.1.4 Propagation of Classification

When a new object or subject is created, the following rules are used to determine its classification:

- Subprocess - The same classification as parent process.
- Batch job - The same classification as the submitting process.
- File - The same classification as the creating parent process.
- Global section - If mapped to a data file (not the page file) the classification of the file; otherwise, the classification of the process.

- Shared logical name tables - The classification of the process.

Note: The system and group logical name tables are not automatically classified by SEVMS. They must be explicitly classified if desired. If they are classified, SYSNAM and GRPNAM privileges do not bypass the mandatory access check.

3.2 File Protection Features

This section describes the file protection features of SEVMS. It contains information about the following topics:

- I/O operations and ODS-2 checks
- Access accountability

3.2.1 I/O Operations and ODS-2 Access Checks

SEVMS mandatory protection checks are made at a variety of levels. These checks occur as a subject initiates the several kinds of operations necessary to gain access to, and perform I/O operations on, an ODS-2 "file" object. These checks are performed in addition to the standard VMS protection checks described in the *Guide to VMS System Security*.

The operations and requirements imposed by SEVMS are as follows:

1 To "initialize" an ODS-2 volume on a device:

- The classification range specified for the volume must be within the user's account classification range (from SYSUAF.DAT).
- The volume classification range must fall within the device classification range.
- The process must have read/write access to device.

2 To "mount" an ODS-2 volume on a device:

- The volume classification range must fall within the device range.
- The process must have read access to the device.
- The process must have read access to the volume.

Note: If the mount is successful, the device's classification is changed to match the classification of the volume. "Read access to the volume" means that the process still has read access to the device after the mount takes place. When the volume is dismounted, the original device classification is restored.

3 To "create" a new file:

- The process must have read/write access to the device.
- The process must have read/write access to the volume.
- If the file is to be in a directory, the process must have read/write access to the directory.
- The process's classification is propagated to the file.

4 To "open" a file for read access:

- The process must have read access to the file.
- The process must have read access to the device.
- If opening the file by name, the process must have read access to all the directories needed to find the file.

5 To "open" a file for write, extend or control access:

- The process must have read/write access to the file.
- The process must have read/write access to the device.
- If opening the file by name, the process must have read access to all the directories needed to find the file.

Note: VMS does not support WRITE ONLY access to a file; write access implies read access.

6 To "read" or "write" to a file:

- When the first read or write operation is performed on a newly opened file, the process must have read or write access to the device. This is an additional check which is done the first time a process reads or writes to a file.

7 To "delete" a file:

- The process must have read/write access to the file.
- The process must have read/write access to the device.
- The process must have read/write access to the directory which contains the file. (That is, the directory pointed to by the file's directory back-pointer.)

8 To "modify" a file's classification:

- The file must be closed.
- The process must be able to open the file for control access (refer to *Item 5* of this section for further information about this topic).
- The new classification of the file must be such that a process running with the current process's privileges and the file's old classification has access to an object with the new classification (i.e. non-privileged processes can only raise a secrecy classification or lower an integrity classification.).
- The new classification of the file must be such that a process running with the current process's privileges and the file's new classification could create a file on the device.
- If accessing the file by name, the process must have read access to all the directories needed to find the file.

9 To "modify" a file attribute (other than classification):

- The process must be able to open the file for control access (refer to *Item 5* of this section for further information about this topic).

3.2.2 Access Accountability

SEVMS augments the information provided by standard VMS *file access* security alarms with information about the classification of the file being accessed. In addition, SEVMS provides the ability to enable alarms of successful and unsuccessful file access by level and/or category.

Refer to the *Guide to VMS System Security* for general information about security alarms and VMS auditing capabilities. See Chapter 6 of this manual for more information on SEVMS auditing capabilities and using SEVMS-specific alarms.

4 Mandatory Access Control and the User

This chapter discusses how mandatory access control relates to the system user.

The following topics are addressed in this section:

- How the user logs into the system
- How the user determines object classification

4.1 Logging into the System

When a user logs into the system, LOGINOUT creates a process for that user. The ARB (access rights block) of the resulting process is determined by the following factors:

- the contents of the authorization file, SYSUAF.DAT
- additional information specified at log in

At log-in time, the user may specify the classification of the process. This specification is validated against the minimum and maximum classifications stored in the user's UAF record according to rules outlined in Section 4.1.1, which follows.

4.1.1 Specification of Session Classification

To specify the security of a session for a user account assigned a classification range, the /SECRECY qualifier is appended to the username given at log-in time. The classification specified with /SECRECY consists of a single classification (not a range).

Note the following example:

```
Username: MARVIN/SECRECY=(LEVEL:BIG_CHEESE,CATEGORY:(CHEDDAR,BRIE))
Password:

 Welcome to VMS . . .
```

The requested classification is compared to the range the account provides. Incorrectly specifying security level and category identifiers, either by using the wrong names, or by attempting values that are out of range for the account, causes a LOGIN failure. In this case, the following message is displayed:

```
User authorization failure
```

The LOGIN failure message is *generic* in the sense that it does not give an explicit reason for the rejection of the attempted log-in. This prevents the leakage of any information to a person making a determined attack on the system.

If no classification is specified by the user when logging in, then the classification of the session defaults to the *maximum* class allowed for that user.

Users of singly classified accounts are always logged in at the classification stored in the authorization file.

Users cannot specify a classification when logging in to a DECwindows session on a VAXstation. Users must always log in at their *default* classification. Furthermore, in this version of SEVMS, users **must** log in to an *unclassified* DECwindows session (i.e. LEVEL = 0, CATEGORIES = NONE). Attempting to log in to an classified DECwindows session will cause an "UNSUPPORTED OPERATION OR FUNCTION" error message to be displayed.

4.1.1.1 Captive Accounts

If a user account is designated as "CAPTIVE" in the user authorization file, then the user cannot log in using the /SECRECY qualifier. Classifications for users whose accounts are specified as CAPTIVE accounts always default to the maximum allowed for the user. If the user attempts to specify a /SECRECY string for a CAPTIVE account, the log in attempt fails.

4.1.1.2 Batch Processes

Under SEVMS, batch jobs run at the classification of the process which submits the job for execution. The classification is stored at the time the job is submitted and is fixed; it is not possible to specify or modify it. When the batch job is started, the user account under which the job is to run must be authorized to log in at the stored classification. If not, a BATCH LOGIN failure results.

4.1.1.3 Network Processes

A network process runs at the classification of the remote process which initiated the network connection. Refer to Section 5.1 of this manual for further information about this topic.

4.1.1.4 Detached Processes

By default, a detached process runs at the *current* classification of the process that creates it. However, the $CREPRC system service can be used to specify another classification when it creates a detached process. Privileges are required to do so. Refer to the "Programming Information" section of the *VMS SES User's Guide* for more information about this topic.

4.2 Displaying Classification of a Process

The classification of a process can be obtained with any of the following commands:

```
$ SHOW CLASS/PROCESS
$ SHOW CLASS/PROCESS/ID = <process-id>
$ SHOW CLASS/PROCESS <process-name>
```

In the above commands:

- <process-id> is a VMS numeric process identification

- <process-name> is a VMS process name

If neither "/ID=" nor "<process-name>" is specified, then the classification of the user's process is returned. To use <process-name>, the process must be in the same group. See "DCL Commands" section of the *VMS SES User's Guide* for more information on SHOW CLASS/PROCESS.

4.3 Classification

The following topics are discussed in the section:

- How to determine object classification
- How to organize classified directories
- How to delete directories
- How to delete files
- How to treat name spaces

4.3.1 Determining Object Classification

A variety of commands are available to determine the classifications of various objects. To view the classification of an object, a user must meet the normal VMS (or non-mandatory) requirements for accessing object information.

For example, a SHOW PROCESS of another process requires WORLD or GROUP privilege.

The classification of objects may be determined with the SHOW CLASS command. The syntax of the SHOW CLASS command is:

```
$ SHOW CLASS[/OBJECT=<object-type>] <object-name>
```

In this command, the <object-type> may be one of the following:

- FILE
- DEVICE
- LOGICAL_NAME_TABLE
- PROCESS
- SYSTEM_GLOBAL_SECTION
- GROUP_GLOBAL_SECTION
- QUEUE

In the previous command, <object-name> designates the object to be examined.

The default <object-type> is FILE.

The following examples illustrate legal SHOW CLASS commands.

```
$ SHOW CLASS MYFILE.DAT
$ SHOW CLASS/OBJECT=DEVICE MTA0:
$ SHOW CLASS/OBJECT=LOGICAL_NAME_TABLE LNM$SYSTEM_TABLE
```

The classification of FILES may also be determined by either of the following commands:

```
$ DIRECTORY/FULL <file-spec>
$ DIRECTORY/SECURITY  <file-spec>
```

Refer to the *VMS SES User's Guide* for more information about the SHOW CLASS command.

4.3.2 Organizing Classified Directories

Accounts which contain classified subdirectories should be organized such that the classification of the directory hierarchy is monotonically increasing. That is, the classification of each directory should dominate it's parent. This is a consequence of the file system requiring read access to intermediate directories when looking up files.

For example, consider the following directory string:

[UNCLASSIFIED.TOP_SECRET.SECRET]

This directory string refers to directories that are classified as UNCLASSIFIED, TOP_SECRET and SECRET respectively.

An attempt to access the following secret classified file from a SECRET process will fail, because the process cannot read the intervening "TOP_SECRET" directory.

[UNCLASSIFIED.TOP_SECRET.SECRET]SECRET.FIL

The one proper way to organize such a directory structure would be as follows:

[UNCLASSIFIED.SECRET.TOP_SECRET]

This arrangement places the SECRET directory in the hierarchy prior to the TOP_SECRET files.

4.3.2.1 Protecting Directory Names

In the example of the previous section, the existence of the "TOP_SECRET" directory, embodied in the file

[UNCLASSIFIED.SECRET]TOP_SECRET.DIR

is *not* concealed from a SECRET process, although such a process cannot access its contents. This is because VMS directory files *are files themselves* and the SECRET process can read a SECRET file.

If the directory name is itself sensitive, then the directory must be "embedded" in an extra directory level which is classified at the same level.

For instance, using the previous example as a starting point, we can embed a TOP_SECRET directory as follows. A new directory named DUMMY is created with a TOP_SECRET classification. The TOP_SECRET subdirectory is then created in the DUMMY directory.

The resulting directory specification would then look like the following:

[UNCLASSIFIED.SECRET.DUMMY.TOP_SECRET]

SECRET processes are now unable to see a directory named TOP_SECRET because they do not have read access to DUMMY.

4.3.2.2 Directory Creation by Privileged Users

Implementing a hierarchy of classified directories is most easily accomplished by a privileged user. A privileged user can directly create the necessary directories and classify them in any way desired.

The following privileges are useful when creating classified directories. The BYPASS privilege by itself is always sufficient. The READALL privilege could be used by a process at system low (the minimum secrecy level in use). The DOWNGRADE privilege could be used by a process at system high (the maximum secrecy level in use).

4.3.2.3 Directory Creation by Unprivileged Users

Unprivileged users can also create such a directory hierarchy, however, it requires several additional steps. Use the following procedure:

1. Log in to an account at the classification of the parent directory.
2. Create the new subdirectory with "CREATE/DIRECTORY".
3. Reclassify the new directory file with "SET CLASS/OBJECT=FILE".
4. Log out, then log in again at the new classification, to access the directory.

The reason this procedure is necessary is that for a process to write a directory, the protection check requires that the directory classification equal that of the process.

Thus, a TOP_SECRET process cannot create a directory such as [SECRET.TOP_SECRET] because it cannot write the subdirectory file name into the SECRET directory.

A SECRET classified process may, however, create a directory called "TOP_SECRET" (with a SECRET classification by default), and then re-classify it upward with the "SET CLASS" command for the later use of the TOP_SECRET process.

4.3.3 Name Spaces

Section 4.3.2.1 discusses a principle that applies to more than just directories. *Names* are not "objects" to SEVMS, and are therefore *not protected* by the classification of the object they represent. Sensitive names may be visible to users, even though the tangible contents of the objects represented by those names may be inaccessible.

Some examples of this are the following:

- Classified files entered into directories protected at a lower classification.
- Secrecy and category "named" identifiers.
- Classified logical name table names cataloged in an unclassified directory of logical name tables, such as the system logical name table.

Since the existence of names cannot be fully hidden, and descriptive names may provide tempting targets, consideration should be given to the use of context-free "nonsense" names. In other words, rather than using the named category "super_secret_widget", the use of an arbitrary name such as "spinach_salad" may be preferable.

Another technique is to create "dummy" objects to contain the actual names of objects. Section 4.3.2.1 explained the how to do this with directories. Similar methods can be used with other objects.

4.4 Printing Files

The services of the SEVMS print symbiont are accessed in the same way as the services of the standard VMS print symbiont. Users queue documents for printing by using the DCL PRINT command as described in the *VMS DCL Dictionary*. Users are able to specify all of the same qualifier options that the standard symbiont accepts.

However, in regard to protection checking, SEVMS operates unlike standard VMS. In VMS, protection checking is performed at the time the PRINT command is issued, and is limited to determining that the user can read the file. In SEVMS, the print symbiont does additional protection checks at the time the file is printed to see that the file's classification falls within, or is equal to, the classification range of the target printer. If the classification of a file that is submitted for printing is less than or greater than the classification range of the target printer, an access violation occurs and the file will not be printed.

As a result, the user might see a NO PRIVILEGE error returned as a job completion status if this print-time check fails.

5 Mandatory Access Control and the Security Manager

This chapter discusses mandatory access control as it relates to the security manager (or system manager).

The topics covered in this section are:

- Security management
- Privileges
- Defining secrecy levels
- Defining secrecy categories
- Establishing account classifications
- Protecting critical objects
- Changing classification
- Enabling and disabling mandatory controls
- BACKUP
- Networks
- MAIL

5.1 Security Management

It is the responsibility of the security manager to configure the system in accordance with the security policy of the site. SEVMS provides many features to enforce this policy, but they will not operate as desired without some thought and effort in designing and implementing the protection scheme.

The duties of the security manager include the following:

- Determining the basic levels and categories, and assigning to them identifiers in the rights database with the AUTHORIZE utility
- Determining classification ranges for each disk volume, and initializing the disks appropriately
- Determining minimum and maximum classifications for each terminal and printer, and setting them at boot time with SET CLASS/OBJECT=DEVICE (in SYSTARTUP.COM, for example)
- Determining minimum and maximum classifications for each user, and adding this information to the user authorization database with the AUTHORIZE utility
- Defining the operational characteristics of, and starting, the secure print symbiont

5.2 Privileges

SEVMS does not remove or limit existing VMS privileges. Users with any of the ALL privileges (as defined in the *Guide to VMS System Security*), can manage to circumvent the mandatory access controls, just as they can bypass the discretionary access controls. Similiarly, users with DEVOUR privileges can consume system resources, those with FILE privileges can bypass file security, and so on.

For these reasons, it is important that the security manager be familiar with VMS privileges and restrict their use.

The privileges which directly affect mandatory access checks are as follows:

- DOWNGRADE - Allows a process to write to a lower secrecy object or to lower an object's classification.
- SECURITY - Allows a process to set a ranged classification and to turn on and off security auditing.
- BYPASS - Bypasses all protection checks.
- VOLPRO - Bypasses volume protection checks.
- UPGRADE - Allows a process to write to a higher integrity object or to raise the integrity of an object.
- READALL - Allows any object to be read.

5.3 Defining Secrecy Levels

This section describes the procedure for defining secrecy levels.

Step 1

The first step in defining secrecy levels is to determine the number of different levels which are required at your site. Secrecy levels are intended to rank data in a strict hierarchical order of sensitivity.

Step 2

The next step in defining secrecy levels is to assign a text identifier to each level. These identifiers may be descriptive, and generally indicate either of the following things:

1 The sensitivity of the data

2 The clearance of the person qualified to access the data

Some examples of identifiers could be the following:

- Department of Defense: UNCLASSIFIED, CONFIDENTIAL, SECRET, TOP_SECRET
- Bank organization: TELLER, MANAGER, VP, BOARD, AUDITOR
- Engineering organization: PROGRAMMER, GROUP_LEADER, PROGRAM_MGR

Step 3

Finally, these identifiers are entered into the rights database with the AUTHORIZE utility. The relevant commands are as follows:

```
UAF> ADD/IDENTIFIER/SECRECY=(LEVEL:<number>) <identifier>
UAF> REMOVE/IDENTIFIER/SECRECY  <identifier>
UAF> SHOW/IDENTIFIER/SECRECY    <identifier>
```

In the above commands:

- <identifier> is a valid VMS identifier text string (but no longer than 29 characters long).
- <number> is the numeric value of the level, which must be an integer in the range of 0-255.
- Synonyms for levels are not allowed.

Note: **Integrity levels are added in a similar manner using /INTEGRITY in place of /SECRECY.**

Generally, once a scheme for secrecy (or integrity) levels is settled upon, it becomes very difficult to change at a later time due to the significant effort required to reclassify all files and disk volumes.

For example, suppose that levels 1 and 2 originally represent SECRET and TOP_SECRET, but are reassigned to levels 2 and 3 to make room for a CONFIDENTIAL classification level 1. Existing files that had originally been assigned to SECRET would then, in effect, have been *downgraded* to CONFIDENTIAL, as would files previously classified at TOP_SECRET to SECRET. Therefore, to correct the classification of the files (and disk volumes) it would be necessary to explicitly reclassify all such objects with SET CLASS.

The effort required to change the classification of files is potentially large. Therefore, secrecy levels must be carefully considered *before* defining them.

Unless you are very certain that the scheme will never be changed, you should leave some gaps between levels for future expansion. (There is no requirement that levels be tightly grouped.)

For example, one might make the following assignments:

```
UAF> ADD/IDENTIFIER/SECRECY=LEVEL:0  UNCLASSIFIED
UAF> ADD/IDENTIFIER/SECRECY=LEVEL:20 CONFIDENTIAL
UAF> ADD/IDENTIFIER/SECRECY=LEVEL:30 SECRET
UAF> ADD/IDENTIFIER/SECRECY=LEVEL:40 TOP_SECRET
```

5.4 Secrecy Categories

This section introduces the topic of secrecy categories. It contains information describing secrecy categories, how secrecy categories are used, and how secrecy categories are set up.

5.4.1 Description

Secrecy categories are intended to assign non-hierarchical attributes to an object. Categories are generally used to compartmentalize information; a subject may possess multiple categories at once.

SEVMS allows the use of 128 secrecy categories. Versions of SEVMS prior to Version 4.6 allowed only 64 secrecy categories. However, limitations are imposed on the use of the new categories because secrecy categories 65 through 128 and integrity categories 1 through 64 use the same fields in some internal data structures. Secrecy and integrity *levels* are not effected. The following list describes some of the limitations which affect the use of 128 secrecy categories. Note that in this list, minimum and maximum classifications of objects with classification ranges are considered to be two separate classifications.

The limitations of 128 secrecy categories are as follows:

- No classification can include both a secrecy category greater than 64 and an integrity category.
- For access control purposes, classifications with secrecy categories greater than 64 are considered to have no integrity categories.
- For access control purposes, classifications with integrity categories are considered to have no secrecy categories greater than 64.
- The 64 secrecy categories used by Versions 4.4 and 4.5 of SEVMS are the same as the first 64 secrecy categories of later versions of SEVMS. No conversions are required.
- Secrecy categories 65 through 128 would be treated as integrity categories by Version 4.4 or 4.5 of SEVMS. **Use of categories greater than 64 is not backwards compatible to Versions 4.4 and 4.5 of SEVMS.**
- Secrecy categories 65 through 128 share the same rights database values as the integrity categories. Therefore, an identifier (specified by the AUTHORIZE ADD/IDENTIFIER command) can only be specified for an integrity category or the related secrecy category greater than 64, but not both. For instance: if RED was defined as integrity category 1 and BLUE was defined as secrecy category 66, then no identifier could be assigned for RED to secrecy category 65 and no identifier could be assigned for BLUE to integrity category 2.
- If no identifier is available for a given category, the system will display (or can be given) the numeric value.
- All 128 secrecy categories and 64 integrity categories can be audited independently.

5.4.2 Defining Secrecy Categories

This section describes how to go about defining secrecy categories.

Step 1

The first step in defining secrecy categories is to determine the number of different categories which are required at your site.

Step 2

The next step in defining secrecy categories is to assign a text identifier to each defined category. These identifiers should be descriptive, and generally indicate either of the following:

- The project which "owns" the data
- The organization of the person qualified to access the data

Some examples of this are:

- Descriptive: DOD, NATO, NOFORN, ATOMIC, INTELLIGENCE
- Cryptic: STAR, VENUS, SCORPIO, JUPITER
- Corporate organization: ENGINEERING, MARKETING, SALES, PERSONNEL

Step 3

These identifiers are entered into the rights database with the AUTHORIZE utility. The syntax used to enter an identifier is illustrated below:

```
UAF> ADD/IDENTIFIER/SECRECY=(CATEGORY:<number>) <identifier>
UAF> REMOVE/IDENTIFIER/SECRECY                  <identifier>
UAF> SHOW/IDENTIFIER/SECRECY                    <identifier>
```

In the above example, note the following points:

- The <identifier> is a valid VMS identifier text string (but no longer than 29 characters long).
- The <number> is the numeric value of the CATEGORY, which must be an integer in the range of 1-128.
- Synonyms for categories are not allowed.

Note: Integrity categories, though unsupported, are added in a similar manner using /INTEGRITY in place of /SECRECY.

Because categories are non-hierarchical, the explicit numeric values associated with them are not as important as with the secrecy levels.

The following examples further illustrate the use of secrecy categories:

```
UAF> ADD/IDENTIFIER/SECRECY=CATEGORY:1 RED
UAF> ADD/IDENTIFIER/SECRECY=CATEGORY:2 ORANGE
UAF> ADD/IDENTIFIER/SECRECY=CATEGORY:3 YELLOW
UAF> ADD/IDENTIFIER/SECRECY=CATEGORY:4 GREEN
UAF> ADD/IDENTIFIER/SECRECY=CATEGORY:5 BLUE
UAF> ADD/IDENTIFIER/SECRECY=CATEGORY:6 INDIGO
UAF> ADD/IDENTIFIER/SECRECY=CATEGORY:7 VIOLET
```

The temptation to assign a user to a long series of categories should be strenuously resisted, as the corresponding secrecy strings may then require the listing of a large number of categories, and consequently be quite difficult to deal with.

Imagine your users having to log into an account with the following string

```
/SECRECY=(LEVEL:SECRET,CATEGORY:(RED,ORANGE,YELLOW,GREEN,INDIGO,VIOLET))
```

Instead, consider the following coarser and simpler string:

```
/SECRECY=(LEVEL:SECRET,CATEGORY:(RAINBOW))
```

Note: **In particular, remember that *continuation lines are not permitted at the LOGIN username prompt*. If a string can't be specified in a single line, it is too long.**

5.5 Changing Classification

An object's classification is changed with the "SET CLASS" command. A non-privileged user can only increase the classification of an object.

The use of UPGRADE, DOWNGRADE, and BYPASS privileges, in relation to changing classification, is described in this section.

5.5.1 UPGRADE Privilege

A process with UPGRADE privilege is allowed to *increase* the *integrity* of an object. Note that UPGRADE privilege is used *only* for integrity.

5.6 Enabling and Disabling Mandatory Controls

The entire mandatory control scheme is controlled by the dynamic SYSGEN parameter "CLASS_PROT".

- When set to "1", mandatory controls are enabled and object classifications are propagated when new objects are created.
- When set to "0", mandatory controls are disabled.

Note: **Disabling mandatory controls with "CLASS_PROT" is not recommended, it causes the following things to happen:**

- **No mandatory access checks are made and only the standard VMS protection checks are enforced.**
- **Objects created from that point on will not be labeled.**

5.6.1 DOWNGRADE Privilege

A process with DOWNGRADE privilege is allowed to *decrease* the *secrecy* of an object. Note that DOWNGRADE privilege is used *only* for secrecy.

5.6.2 BYPASS Privilege

A process with BYPASS privilege circumvents the entire security scheme and may perform any operation without restrictions. While BYPASS overrides all discretionary (UIC and ACL) and mandatory (SEVMS) protection checks, it does not bypass checks for privileges unrelated to access.

5.7 Protecting Critical Objects

The task of classifying objects in the secure environment is the most time consuming part of configuring the mandatory environment. Conceptually, object classifications fall into the following two types:

- *Static*
- *Dynamic*

Static classification involves "permanent" objects such as disk volumes and files. The data structure that contains the classification block resides on disk. Hence, when these objects are classified, the effect is permanent and need only be done once.

Dynamic classification involves objects whose classification blocks reside only in VAX memory, in databases for devices such as terminals, tape drives, and so on. Such objects must be re-classified at every bootstrap in the site-specific system startup procedure provided by the system manager.

Setting Object Classification

The classification of objects is set with the "SET CLASS" command. The syntax of the SET CLASS command is as follows:

```
$ SET CLASS[/OBJECT=<object-type>] <object-name>
```

In the above command:

- The <object-type> may be one of the following:
  - FILE
  - DEVICE
  - LOGICAL_NAME_TABLE
  - QUEUE
  - SYSTEM_GLOBAL_SECTION
  - GROUP_GLOBAL_SECTION
- The <object-name> designates the object whose classification is to be set.
- The default <object-type> is FILE.

For example, the following commands are legitimate SET CLASS commands:

```
$ SET CLASS/SECRECY=(LEVEL:SECRET) MYFILE.DAT
$ SET CLASS/OBJECT=DEVICE/SECRECY=(LEVEL:SECRET,CATEGORY:RED) MTA0:
```

Several commands synonymous to SET CLASS are available. These are specified in the following list:

```
$ SET DEVICE/CLASS <dev-name>
$ SET FILE/CLASS <file-spec>
$ SET DIRECTORY/CLASS <dir-spec>
```

5.7.1 Classifying Devices

SEVMS supports classification of the following devices: terminals, VAXstation display devices, disk devices, magnetic tape devices, mailboxes, and printers. Classification of other devices may have no effect, or may result in system deadlock or failure. This could occur because the software supporting the device may not properly deal with classifications.

Devices are classified with the SEVMS "SET CLASS/OBJECT=DEVICE" command. Device classifications reside only in memory, and are reset at boot time. Therefore, the best place to classify devices is in SYSTARTUP.COM. The syntax of the command to do this is:

```
$ SET CLASS/OBJECT=DEVICE[/SECRECY=<class-string>] <dev-name>
```

Processes attempting to access such a device are subject to a mandatory access check. If the classification of the process is not within the classification range of the device, the assign fails.

5.7.1.1 Terminals

In addition to the access check described in Section 5.7.1, if a terminal is classified, the following protection is provided:

- A log-in fails if the classification requested (or defaulted to) is outside the minimum and maximum limits of the terminal.

5.7.1.2 VAXstations

SEVMS is supported on VAXstations. For this version of SEVMS, the entire VAXstation display should be considered a single output device. No changes have been made in SEVMS software to increase the granularity of mandatory controls to individual windows; however, other SEVMS features may be useful on a VAXstation.

5.7.1.2.1 VWS Window Manager Support

SEVMS provides no unique protection in relation to the window manager; however, VWS can be configured in ways compatible with mandatory controls. For instance, the main menu can be set up to allow only automatic login. Then, after logging in normally the first time, the user can only log into the same single classification account by way of the window manager.

In the SEVMS environment, this means that if the first login were into a classification "A" account, all additional logins would be into a classification "A" account.

Windowing

The ability to label template devices allows the template for terminal emulation windows (WT: and TK:) devices to be labeled. This ability can be used to limit access to terminal emulation windows and log-ins. Aside from that, virtual terminals can be labeled in the same way that normal devices are labeled.

5.7.1.2.2 DECwindows

SEVMS allows only unclassified users to log in to DECwindows sessions.

Classification of DECterm terminal emulation windows is not supported.

5.7.1.3 Disk Devices

The VMS MOUNT command will fail if the classification of the process requesting the mount is outside the minimum and maximum limits of the physical disk device.

In addition to the access check described in Section 5.7.1, if the volume is being mounted as an ODS-2 labeled volume, the classification range of the volume itself must fall within that of the physical drive, or the mount fails. If the mount is successful, the minimum and maximum limits of the volume replace that of the device, for as long as the volume is mounted.

5.7.1.4 Tape Devices

ANSI tape volumes do not contain classification information. However, it is still possible to protect a tape by classifying the device it is on. In this case, the VMS MOUNT command fails if the classification of the process requesting the mount is outside the minimum and maximum limits of the tape device. Tapes created with BACKUP contain file classification information within their save sets. BACKUP is discussed in Section 5.10.

5.7.2 Classifying Volumes

SEVMS allows a FILES-11 (ODS-2 only) volume to be initialized to a classification range. Once initialized, this information is checked at mount time against the classification range of the device on which the disk is mounted. Only volumes which have a classification range entirely contained by that of the drive can be mounted. Attempts to mount a volume on a drive which has a lesser classification range than the volume will fail.

When a disk volume is successfully mounted, the mandatory access controls of SEVMS prevents files from being created on the volume that fall outside the range of the volume.

Initialization of a volume is accomplished with the SEVMS "INITIALIZE" command. The syntax of this command is shown in the following example.

```
$ INITIALIZE[/SECRECY=<class-string>][standard initialize qualifiers]-
_$ <device-spec> <volume-name>
```

In this example, the default for the classification is the user's authorized range.

Volume initialization is the only means of attaching a classification range to a disk volume. Attaching a classification range to an existing volume requires the use of the BACKUP utility.

To do this, use the following procedure:

1 Save the volume to tape using BACKUP.

2 Initialize the disk volume with the appropriate classification range.

3 Restore the tape to the disk using BACKUP, specifying /NOINIT.

4 Classify the files appropriately.

Note: **There is a second, less obvious, reason for following this procedure. The classification of a file is stored in the file's header. Existing VMS systems do not set aside room in file headers for this information. Attempts to classify such files result in "header too small" errors. The backup utility provided with SEVMS re-arranges the file headers to make room for the classification block.**

Once the CLASS_PROT SYSGEN parameter has been turned on by the installation of SEVMS, all files created subsequently have space set aside for the classification block.

Tape volumes can be initialized only as unclassified.

5.7.2.1 Protecting Files

When a file is created, it inherits, by default, the current classification of the creating process.

The user can change the classification of an existing file with the SEVMS "SET CLASS" DCL command, subject to the following restrictions:

- The file must be closed.
- The user must have read and write access to the file.
- The file's NEW classification must be between the minimum and maximum classification of the volume on which it is located.
- A non-privileged user can only increase the classification of a file.

The syntax of the SEVMS "SET CLASS" command is illustrated in the following example:

```
$ SET CLASS[/SECRECY=<class-string>]  <file-spec>
```

In this command:

- <file-spec> is a general VMS file specification.
- <class-string> has been previously defined.

The classification of a file is stored in its file header. Existing VMS systems do not set aside room in the file headers for this information. Attempts to classify such files result in a "header too small" error. The BACKUP Utility provided with the SEVMS kit re-arranges the file headers to make room for the classification.

To set the classification of all files on the system created prior to the installation of SEVMS, follow the same procedure used to protect volumes:

1 Save the files to tape using BACKUP.

2 Initialize the disk volume with the appropriate classification range.

3 Restore the files to the disk, specifying BACKUP/NOINIT.

4 Classify the files appropriately with SET CLASS/OBJECT_TYPE=FILE.

5.7.2.1.1 Protecting SYS$SYSTEM

Many of the executable images in SYS$SYSTEM require that every user on the system have read access to them. As such, these files need to be designated as *UNCLASSIFIED*.

This is not limited to directory SYS$SYSTEM or to files associated with VMS. In general, ALL files that require read access by UNCLASSIFIED users need to be set down to UNCLASSIFIED.

5.7.2.2 Protecting Directories

Directories are created at the same classification as the creating process. Normally, directory files do not need reclassification. The "SET CLASS/OBJECT_TYPE=FILE" command can be used if reclassification is necessary.

All the files in a directory should have the same classification as the directory itself. The only exception to this rule is that a directory may contain a sub-directory with a higher classification. Note, however, that SEVMS does not currently prevent a file from being cataloged in a directory of a lower classification.

Because a user must have access to all directories on a path to a file, subdirectories should increase monotonically in classification from the root.

5.7.3 Classifying Global Sections

When created, global sections are single-level objects.

Group and system global sections are protected with the following "SET CLASS" commands:

```
$ SET CLASS/OBJECT=GROUP_GLOBAL_SECTION-
_$ [/SECRECY=<class-string>] -
_$ <global-section-name>

$ SET CLASS/OBJECT=SYSTEM_GLOBAL_SECTION-
_$ [/SECRECY=<class-string>] -
_$ <global-section-name>
```

5.7.4 Classifying Logical Name Tables

All logical name tables, except for default system tables (LNM$SYSTEM) and default group tables (LNM$GROUP_nnnnnn), are created as single level objects whose classification is the same as the process that creates them. As with all objects, SECURITY privilege and control access are needed to reclassify logical name tables with a range.

Default system tables and default group tables are created without any valid classification. Therefore, any process with appropriate privileges (such as SYSNAM or GRPNAM privilege, respectively) can write to these tables, and any process (on the system and in the group, respectively) can read them. If desired, these default tables can be classified during system startup by use of the SET CLASS command. They can be classified to either a single level or to a range.

A group logical name table is normally not created until the first time a member of the group logs in. Group tables can be classified after such a log in takes place (for instance, in a SYLOGIN.COM procedure), or, a program can be run to create and classify the tables during system startup. Because GRPNAM privilege is needed to write into group tables, it may not be necessary to classify them at sites where the privilege is not given to users.

System logical name tables can only be written to by users with SYSNAM privilege. At many sites, this will provide adequate protection. If desired, classification of system tables can be accomplished during system startup A classification of (LEVEL=0) is recommended.

Logical name tables are classified with the following "SET CLASS" command:

```
$ SET CLASS/OBJECT=LOGICAL_NAME_TABLE-
_$ [/SECRECY=<class-string>] -
_$ <logical-name-table-name>
```

5.7.5 Classifying Queues

Mandatory access classification labels on queues are supported by SEVMS. However, SEVMS only uses the label to control access to the queue itself, including submission of jobs. Mandatory access controls do not control access to entries already on queues.

Submission of a job to a queue is considered a WRITE operation. A non-privileged process cannot submit a job to a queue unless the process' classification dominates the queue's classification. But, once a job has been submitted to a queue, the entry can be shown, modified, or deleted - subject only to the applicable discretionary access controls. For example, an unclassified user who issues a SHOW QUEUE command on a classified queue will see any job to which he has discretionary READ access; no mandatory access control check will be applied.

Operations that require EXECUTE access to a queue also require mandatory WRITE access. However, if OPER privilege or access to a job overrides need for EXECUTE access, the mandatory access check is not performed. For example, if a process has DELETE access to a job, a mandatory protection check will not be made if the job is deleted by the DELETE/ENTRY command.

The protection provided on queues described in this section applies to all queues. While it applies to the submission of jobs to print queues by the PRINT command, it does *not* alter the protection checks performed by the SEVMS secure print symbiont.

5.7.6 Classifying Mailboxes

Under SEVMS, the $CREMBX system service sets the classification of a newly created mailbox to be equal to that of the creating process.

Consequently, existing programs which create mailboxes should not have to be modified.

A non-privileged process can "write up" to a mailbox of higher classification if it has WRITE access.

In versions of SEVMS prior to Version 4.7, when an attempt was made to write to a mailbox device, the mandatory control access check required that the accessor have both READ and WRITE access to the device. This prevented a non-privileged process from writing to a mailbox whose classification was greater than that of the process, since the non-privileged process did not have READ access to the mailbox.

In this version of SEVMS, an accessor needs only WRITE access to write to a mailbox device. Therefore, a non-privileged process can "write up" to a mailbox of higher classification.

A mailbox may be protected explicitly with the "SET CLASS/OBJECT=DEVICE" command.

The format of this command is as follows:

```
$ SET CLASS/OBJECT=DEVICE[/SECRECY=<class-string>] <dev-name>
                        -OR-
$ SET DEVICE/CLASS[/SECRECY=<class-string>] <dev-name>
```

5.8 Classification and Interactive Log-ins

This section describes how SEVMS handles interactive log-ins in the classified environment.

When a user logs into an SEVMS system, a new process is created. For the log-in attempt to be successful, SEVMS requires that this new process meets the following criteria:

- It must have a classification within the user's authorized range.
- It must have read and write access to the user's terminal.

If these two conditions are not met, the log-in attempt will fail.

The classification of the user (assigned by means of AUTHORIZE) and the classification of the terminal device (set by means of the SET CLASS/OBJECT_TYPE=DEVICE command) determine whether or not the above conditions are met. But, in addition to these criteria, the user's default privileges are also important factors.

In addition, SEVMS provides a means to classify the pseudodevices created when users log in by means of a terminal server or DECnet.

5.8.1 Classifying Accounts

Once you have defined the classification levels and categories which are to be used on your system, you have to create user accounts, and specify the classification of each user. What this really means is that you must specify the classification of a VMS process which will execute on behalf of the user.

The UAF Record for each user contains a *minimum* and *maximum* classification. The minimum specifies the lowest level and smallest set of categories for processes that the user is allowed to create; the maximum specifies the highest level and largest set of categories for processes that the user is allowed to create.

Once created, however, a process has only *one classification* associated with it. Process classification is established *only* at process creation and cannot be dynamically changed. When a user logs in, the user associates classification with a process that lasts until the user logs out.

Classified users are added to the authorization database with the SEVMS AUTHORIZE utility. This utility is an extension of the standard VMS AUTHORIZE utility. A new user is added with the same commands as in standard VMS, plus new qualifiers to specify *secrecy strings* for the user.

SEVMS uses the same syntax for specifying or reporting secrecy strings in all its utilities, including SET and SHOW. This syntax is shown in the following table.

/SECRECY Qualifier Syntax

```
secrecy-qualifier : = /SECRECY = (class-string)

                      { LEVEL = level-range |                          }
class-string : =      { CATEGORY = category-range |                     }
                      { LEVEL = level-range,                           }
                      {       CATEGORY = category-range                }

                      { level |                                        }
level-range : =       { (MAXIMUM:level) |                              }
                      { (MINIMUM:level, MAXIMUM:level)                 }

                      { category-list |                                }
category-range : =    { (MAXIMUM:(category-list)) |                    }
                      { (MINIMUM:(category-list),                      }
                      {     MAXIMUM:(category-list))                   }

                      { category |                                     }
category-list : =     { (category [, . . . ])                          }
```

Notes:

If not entered, minimum ranges default to 0 (level), or none (category).
Colons (:) and equals signs (=) can be used interchangeably in the qualifier.
If MAXIMUM is specified for a single level object, it is ignored.

An example of a command adding a new *singly classified* user would be as follows:

```
UAF> ADD JUSTIN/UIC=[PROJX,71]/DEV=USER$DISK:-
_UAF> /DIR=[JUSTIN]/OWN="NANCY JUSTIN"-
_UAF> /ACCOUNT=PROJX/PRIV=(TMPMBX,NETMBX,GRPPRV)-
_UAF> /SECRECY=(LEVEL:GRP_LEADER,CATEGORY:(STAR,VENUS))
```

An example of a command modifying an existing account to provide a *classification range* would be as follows:

```
UAF> MODIFY JUSTIN -
_UAF> /SECRECY=(LEVEL=(MIN:PROGRAMMER,MAX:GRP_LEADER), -
_UAF>             CATEGORY=(STAR,VENUS))
```

The /SQn qualifier can also be used to enter long classification strings. Refer to the SET CLASS command description in the "DCL Commands" section of the *VMS SES User's Guide* for information about this qualifier.

Note: When secrecy strings are given to an SEVMS facility, they must *completely specify* the desired classification, because the given secrecy string *completely replaces* the previous object classification. You cannot, for example, "add" a single secrecy category to a file without completely specifying the *entire final secrecy string*.

5.8.1.1 Setting Classifications for the SYSTEM Account

The SYSTEM account on a VMS system is created not only for the personal use of the system manager, but for many other purposes as well. The SYSTEM account is used for system maintenance and operation functions on a daily basis. As such, it possesses many privileges. Furthermore, the SYSTEM account has a special role in the VMS bootstrap sequence (for example, starting up other system processes). Usually, many unprivileged users require read access to files created by the system manager. Consequently, it is important that the account itself, as well as files associated with the account, be created at a classification of (LEVEL = 0, CATEGORIES = NONE).

If it is necessary to create a privileged account, it is recommended that you set up a separate account (for example, USERNAME: SYSTEM_HIGH) whose default log-in directory is some subdirectory of SYS$MANAGER:. This enables the system manager to log in at the classification desired, while preserving the general purpose nature of the SYSTEM account.

5.8.2 Classifying Terminals Connected by Serial Line Interfaces

Terminals which are directly connected to VAX systems by means of asynchronous serial line interfaces are classified with the SET CLASS/OBJECT_TYPE = DEVICE command. Refer to Section 5.7.1 of this manual for more information about classification of devices.

These interfaces are connected to the VAX system with multiplexers (such as DZV11s or DMB32s), system console terminals, and other serial line interfaces that are supported by VAX processors. See the *VMS Operating System Software Product Description* for more information about supported interfaces.

5.8.3 Classifying Terminals Connected by DECnet

Users can log into SEVMS systems interactively through DECnet by using the SET HOST command. When an SEVMS system receives such a request, a pseudodevice with the name RTAn: (where n is a unique unit number) is created. SEVMS assigns a classification to this device based upon the classification of the process initiating the SET HOST. Refer to Section 5.9.6 of this manual for further information.

5.8.4 Classifying Terminal Server Ports

There are three kinds of LAT ports supported by VMS:

- Interactive ports, which are used for interactive sessions
- Applicaion ports, which are used to support non-interactive devices such as printers
- Dedicated ports, which are used to give control of a port to an application program.

SEVMS provides the means to associate classifications with all three kinds of ports. This is described in the following sections.

5.8.4.1 Interactive Ports

When a connection is received by an SEVMS system from a terminal server (such as a DECserver 200), the system creates a pseudodevice with the name LTAn: (where n is a unique unit number). SEVMS provides a means to classify this pseudodevice - based upon the terminal server's name, and the port name on the terminal server. The assignment of a classification to a particular port, or to all ports, on a server is accomplished with the SET CLASS/SERVER command. This command causes the SEVMS system to store the assigned classification in a database file, named SYS$SYSTEM:SEVMS$LOGIN_CLASS.DAT. Refer to the "DCL Dictionary" section of the *VMS SES User's Guide* for information about the use of this command.

When an SEVMS system receives a CONNECT request from a terminal server, the following things happen:

1 The SEVMS system checks its database file to find a classification assignment for the server port initiating the request.

2 If a classification for the port is found, the LTAn: created is given that classification.

3 If a classification for the port is not found, the classification of LTA0: is used instead. (LTA0: is a template device and is classified by the SET CLASS/OBJECT_TYPE=DEVICE command.)

The steps that an SEVMS system performs to classify an LTAn: terminal device are summarized in the following table:

If this condition:	Then SEVMS will:
If there is no database file (SET CLASS/SERVER has never been used)	Use the classification of LTA0:
If there is a database file	Look there for a classification for the specific port on the server
If a classification for the port is found in the database	Use that classification
If a classification for the specific port on the server is not found in the database	Look for a classification for the server
If a classification for the server is found in the database	Use it
If a classification for the server is not found in the database	Use the classification of LTA0:

Notes:

- The server name and port name are sent to the system by the terminal server. The names sent by the server to the system can be displayed by the using a DCL SHOW TERMINAL LTAn: command.

- The database associating servers, server ports, and classification is kept on the SEVMS system and is used only by that particular node or cluster. Other SEVMS systems need to maintain their own databases.
- Because a server port name is not provided to the SEVMS system until a CONNECT is issued on the terminal server, the SET CLASS/SERVER command does not check for the existence of a server or port.
- Some older terminal servers (DECserver 100s) may not provide server name and port name information to the system. This can be checked by the using a DCL SHOW TERMINAL LTAn: command.

5.8.4.2 Application and Dedicated Ports

Application and dedicated ports can have classifications associated with them after they are created by use of the SET CLASS/OBJECT = DEVICE command.

The following example illustrates this:

```
$ RUN SYS$SYSTEM:LATCP
LCP> CREATE PORT LTA103:/APPLICATION
LCP> EXIT
$ SET CLASS/OBJECT=DEVICE LTA103:/SECRECY=(LEVEL:1)
```

The classification of interactive ports (on the same server) as application or dedicated ports will have no effect on the application or dedicated ports.

For further information about creating ports, refer to the VMS *LATCP Manual*. For more information about setting device classification, refer to the *VMS SES User's Guide*.

5.8.5 Privileges and Log-in Classification

Because the user's default privileges affect whether the created process will be able to read or write the terminal device, default privileges affect what classifications the user can login at. Specifically, a user with the following privileges can do the following things:

- BYPASS privilege - user can log in to any terminal regardless of classification.
- READALL privilege - user can log in to a terminal whose secrecy classification dominates the requested secrecy classification and whose integrity classification is dominated by the requested integrity classification.
- DOWNGRADE privilege - user can log into a terminal whose secrecy classification is dominated by the requested login classification.
- UPGRADE privilege - user can log into a terminal whose integrity classification dominates the requested integrity classification.

A user's default privileges are specified in AUTHORIZE with the /DEFPRIVILEGES qualifier.

5.9 Networks in the Classified Environment

This section provides information about networking in the SEVMS environment.

The following topics are discussed in this section.

- Session control
- DECnet connections between SEVMS and non-SEVMS nodes
- Restricting network access
- Network file transfers

Modifications have been made to DECnet so that it is easier to connect SEVMS systems together by means of DECnet without compromising the mandatory control policy. The modifications are in the area of session control.

These changes do not make the actual *physical* network more secure. Some sort of physical security is still needed to protect the data on the network and to control unauthorized access. Also, the individual nodes on the network must themselves be secure from tampering with their network software. Refer to the *VMS System Management Subkit* for further information about making your system's network software secure.

5.9.1 Session Control

Under SEVMS, when a process requests a connection to a remote node, the classification of that process is sent to the remote node along with other DECnet-relevant information. The classification is added to the connect request message by the system; users are not able to specify or modify the classification. This classification is then used on the remote node (if running SEVMS) in selecting a target process to receive the connection.

Under DECnet, there are two types of processes which can receive a connection:

- A process which is a network server *(NETSERVER)*
- A process which has declared itself to be a network object *(DECLARED OBJECT)*

Refer to the *VMS Networking Manual* for further information about this topic.

When a connection request is received by a DECnet node, the DECnet software determines which process is to be the target of the connection, and whether that process is a DECLARED OBJECT.

If the target process is **not** a DECLARED OBJECT, then a NETSERVER process is used, and SEVMS ensures that the NETSERVER process runs at the same classification as the process that initiated the connection request. If the user account under which the NETSERVER process is to run is not authorized for the connect initiator's classification, then the connection request is rejected. The same error message is returned as would be if the connect request contained an invalid password. The attempt is also auditable, on the target node, as a failed NETWORK LOGIN event.

If the process that is to receive the network connection is a DECLARED OBJECT, things work differently. Under DECnet, a DECLARED OBJECT is one that can accept multiple incoming connections (i.e. connections from various processes) concurrently. A DECLARED OBJECT is also expected to do its own access control checking. REMACP, the process that handles "SET HOST" requests, is one example of a DECLARED OBJECT. Under SEVMS, when a connection is requested to a DECLARED OBJECT, the classification of the process requesting the connection is compared to the classification of the DECLARED OBJECT process. If the classifications match, then the connection request is passed to the DECLARED OBJECT (which can then accept or reject the connection). If the classifications do NOT match, the connection is rejected unless the DECLARED OBJECT possesses the DOWNGRADE and BYPASS privileges. If it does possess those privileges, the connection is passed to it.

5.9.2 DECnet Connections Between SEVMS and Non-SEVMS Nodes

Under SEVMS, a DECnet connection can be made between SEVMS (nodes running SEVMS) and non-SEVMS (nodes not running SEVMS) nodes. This is possible because SEVMS treats a process running on a non-SEVMS node and an unclassified process running on an SEVMS node in the same manner. When a connection request from a non-SEVMS node is received by an SEVMS node, the SEVMS node considers the connection as UNCLASSIFIED unless that node has had a classification associated with its NOCLASS links. Refer to Section 5.9.7 for information about associating classifications with nodes. Similarly, when an SEVMS node sends a connection request from an unclassified process, the request contains no classification information. Therefore, connection requests received from non-SEVMS nodes are identical to connection requests received from UNCLASSIFIED processes on SEVMS nodes.

Connection requests sent by non-SEVMS nodes are identical to requests sent from *unclassified* processes on SEVMS nodes. Connection requests sent by non-SEVMS nodes are **not** identical to requests sent from *classified* processes on SEVMS nodes.

5.9.3 DECnet Proxy Access

Generally, file access over the network requires a user to specify authorization information in the form of an access control string, "username password", in the remote file specification. DECnet proxy access allows a process running on one node to access files on another node without specifying a username and password in the access control string. There are two types of proxy access: DEFAULT and NON-DEFAULT. For default proxy access, no access control information is specified; for non-default proxy access, the user specifies a username, but does not specify a password, in the access control string.

There are several factors which influence the way that proxy access works in SEVMS. As mentioned previously, SEVMS treats a process running on a non-SEVMS node as an UNCLASSIFIED process (unless the node has an associated classification). Also, SEVMS passes classification information as

a part of the access control string. Furthermore, the way that DECnet-VAX processes proxy requests affects SEVMS proxy access. The combination of these factors result in the following statements about proxy access in an SEVMS environment:

- Classified processes that are running on an SEVMS node CANNOT gain access to a remote non-SEVMS node using DEFAULT proxy access.
- Classified processes that are running on an SEVMS node CAN gain access to a remote non-SEVMS node using NON-DEFAULT proxy access.
- Unclassified processes that are running on an SEVMS node can gain access to a remote non-SEVMS node using either DEFAULT or NON-DEFAULT proxy access.
- Processes running on a non-SEVMS node can gain access to a remote SEVMS node by using either DEFAULT or NON-DEFAULT proxy access.
- Proxy access between two SEVMS nodes can occur at any classification and can involve either DEFAULT or NON-DEFAULT proxy access.

Further information about proxy access can be found in the VMS System Management/Networking documentation in the *Guide To DECnet-VAX Networking* and the *Networking Manual.*

5.9.4 Restricting Network Access

A local system housing sensitive data may be configured to reject attempts to form inbound connections from some or all remote systems on the network. The effect is to defeat attempts from remote "untrusted" systems to access the local classified system, while retaining the ability of the local system to generate outbound requests accessing any remote network node.

Network Control Program (NCP) *executor default access* attribute controls this behavior on a global basis, while the node specific *access* attribute can override the executor default for designated remote systems.

More precise control can be established on a node-by-node basis by using the SEVMS capability to associate classifications with remote nodes. This capability allows communications (by way of DECnet) to and from another node to be restricted to processes running at either a single classification or within a defined classification range. This makes it possible to limit access from an SEVMS node to non-SEVMS nodes. For example, it could be established that only UNCLASSIFIED connections are allowed to an untrusted node, or only SECRET connections to a secret node. This capability is described in Section 5.9.7.

Note that this does not absolutely prevent the downgrade of information on the local system. The local system may still potentially "push" classified data out to untrusted network nodes if access to those nodes has not been disabled or limited. (Untrusted nodes are nodes which are not running SEVMS and nodes which are not under the control of the local security manager).

5.9.5 Network File Transfers

SEVMS does not support the transport of classification labels for files when they are sent across a network. Consequently, when a file is received at the destination, the original file classification is lost.

However, if the target system is also running SEVMS, the files transferred receive a classification identical to that of the process that initiated the transfer (since SEVMS ensures that the NETSERVER process on the remote end of a DECnet link has a classification equal to that of the local initiating process). If it is not possible for the NETSERVER process to be logged in at the same classification as the initiating process, the file transfer fails.

If a file transfer occurs from a non-SEVMS node to an SEVMS node, the files are set to UNCLASSIFIED on the target SEVMS node, unless the non-SEVMS node has a NOCLASS link classification associated with it.

5.9.6 Remote Terminal Sessions

The DCL command "SET HOST" is used to initiate a terminal session on a remote node in the network from a terminal connected to the local node. (Refer to the *VMS DCL Dictionary* for information about the SET HOST command.) This remote session is logged in on an "RT" device (e.g. RTA1) on the remote node.

If the remote node is running SEVMS, then that RT device is set to the same classification as the process on the local node from which the SET HOST was issued. As the user attempts to log in to some account on the remote node, the classification of that account is checked against the classification of the RT device, just as is done for an interactive login.

The SET HOST attempt will fail if the classification and/or privileges of the specified account on the remote node do not permit a login at the classification of the RT device.

There are two means of restricting remote terminal sessions - classifying the RTA0: device, and associating a remote link classification with the remote node. These methods are described in detail in the following sections.

5.9.6.1 Classifying RTA0:

In addition to forcing the classification of new RT devices to that of the remote, initiating process, SEVMS provides a capability to further restrict the creation of remote terminal sessions. There is a template device, RTA0, that is used in creating the new RT devices as SET HOST requests are received. (RTA0 is never used as a remote terminal, but merely as a template in creating new remote terminal devices.) By default, RTA0 has no valid classification. However, if a classification is associated with RTA0 (via the SET CLASS command), then only remote terminal sessions whose classifications fall within the range of RTA0 are accepted.

For example, if the following command is issued:

```
$SET CLASS/OBJECT=DEVICE/SECRECY:(LEVEL:0,CATEGORY:0) RTA0:
```

Any incoming requests from remote processes (other than unclassified processes) are rejected.

On a given node, the classification of RTA0 **only** affects SET HOST requests which are **received** by that node— it does *not* restrict *outgoing* SET HOST requests.

5.9.6.2 Associating Remote Link Classifications

Remote sessions can also be restricted on a node-by-node basis by means of the SET CLASS/NODE/LINK=REMOTE command. This command restricts incoming requests. Outgoing requests are limited by the SET CLASS/NODE/LINKS=OUTGOING command. This topic is discussed in Section 5.9.7 and in the *VMS SES User's Guide* "DCL Commands" section.

5.9.7 Associating Classifications With Nodes

SEVMS provides the ability to restrict access to and from other nodes in the session control layer of DECnet. These restrictions can be established by the security manager on a node by node basis. The security manager uses the SET CLASS/NODE command to associate classifications (or classification ranges) with other nodes. This command establishes classifications which are associated with specific nodes, for incoming and outgoing logical links.

Some of the advantages that can be derived from this capability are mentioned below.

- It enables an SEVMS system to associate a classification with specific non-SEVMS systems. Therefore, processes running on a non-SEVMS node need not be treated as unclassified by the SEVMS.
- It provides a means to limit DECnet connections to specific nodes to specified classification ranges.
- It provides the means to limit remote terminal (SET HOST) access from specific nodes to specified classification ranges.

The SET CLASS/NODE command has several keywords which are used to associate classifications with various types of logical links from specified nodes. The SHOW CLASS/NODE command can be used to show the classification associations which are established by the SET CLASS/NODE command. Refer to the "DCL Commands" section of the *VMS SES User's Guide* for details about using these commands.

Examples

The following examples illustrate associating classifications with nodes using the SET CLASS/NODE/LINK command.

Assume that NOTSES is a non-SEVMS node. To treat NOTSES as if all users on it were SECRET, the following commands can be issued:

```
$  SET CLASS/NODE/LINK=NOCLASS/SECRECY=(LEVEL:SECRET) NOTSES
$  SET CLASS/NODE/LINK=OUTGOING/SECRECY=(LEVEL:SECRET) NOTSES
```

Assume PINK is another SEVMS node, but only connections to and from processes with category RED are to be allowed to it over DECnet. This limitation can be established by the following commands:

```
$  SET CLASS/NODE/LINK=OUTGOING/SECRECY=(LEVEL:(MIN:0,MAX:255),CATEGORY=RED) PINK
$  SET CLASS/NODE/LINK=INCOMING/SECRECY=(LEVEL:(MIN:0,MAX:255),CATEGORY=RED) PINK
$  SET CLASS/NODE/LINK=REMOTE/SECRECY=(LEVEL:(MIN:0,MAX:255),CATEGORY=RED) PINK
```

Assume ORANGE is another SEVMS node, only connections to and from processes with category RED are to be allowed to it over DECnet (as in the example above), but SET HOST will be allowed only to accounts at level 0, category PINK. This limitation can be established by the following commands:

```
$  SET CLASS/NODE/LINK=OUTGOING/SECRECY=(LEVEL:(MIN:0,MAX:255),CATEGORY=RED) PINK
$  SET CLASS/NODE/LINK=INCOMING/SECRECY=(LEVEL:(MIN:0,MAX:255),CATEGORY=RED) PINK
$  SET CLASS/NODE/LINK=REMOTE/SECRECY=(LEVEL:0),CATEGORY=RED PINK)
```

To restrict SET HOSTs from node YELLOW to CONFIDENTIAL logins, enter the following command:

```
$  SET CLASS/NODE/LINK=REMOTE/SECRECY=(LEVEL:CONFIDENTIAL) YELLOW
```

To restrict all DECnet activity in a cluster to UNCLASSIFIED processes, except SET HOST, the following commands can be issued for each node in the cluster.

```
$  SET CLASS/NODE/LINK=OUTGOING/SECRECY=(LEVEL:0) node-name
$  SET CLASS/NODE/LINK=INCOMING/SECRECY=(LEVEL:0) node-name
```

Note: **Classifications associated with nodes should be consistent across the network. Each SEVMS system uses it's own classification database to limit the ability its processes to establish connections with other systems. There is no "network wide" database and no exchange of node classification between nodes. Limitations established by the SET CLASS/NODE command apply to that system only.**

5.10 BACKUP Utility in SEVMS

New standalone and online BACKUP images are provided with SEVMS. These images *preserve* file classification labels present on ODS-2 disk volumes. *Standard BACKUP images do not preserve file classification labels.* This will likely be discovered if you continue using an old copy of standalone BACKUP to save your disks, resulting in the loss of your classification labels.

It is essential to rebuild standalone BACKUP and *discard* any old copies of BACKUP to prevent this catastrophe.

5.11 Mail in the Classified Environment

MAIL functions properly when the classification of the sending process is dominated by the classification of the receiver's default MAIL directory.

Attempting to send mail to a lower classification results in a protection violation.

For mail to be sent from one DECnet node to another, the account that the MAIL object runs in, on the receiving node, must be set up so that the following conditions are met - in addition to any normal VMS requirements:

- The account must be authorized to log in at the classification of the remote process.
- The account must have write access to its default directory when running at the classification of the remote process.

If you wish to allow any remote process to send to any local mail directory which has a classification that dominates that of the remote process, then the DECnet MAIL object should be set up to run in its own specific default account. That account should be set up with the following characteristics:

- It should not be shared with any other object.
- It should have a default directory which is unclassified.
- It should have DOWNGRADE and a default and authorized privilege
- It should be authorized to run at any classification.

The use of a default DECnet account is not recommended (see *Guide to VMS System Security* Section 8.5), but if it is used, it should not be given any privileges. It is better to use individual accounts for each network object, and, in particular, an individual account for the MAIL object, so it can be given DOWNGRADE privilege and authorized for all classifications. Of course, if you do wish to restrict mail so that it can only be sent from remote processes with classifications that are the same as that of the mail account's default directory, you need not give the account DOWNGRADE privilege. Likewise, you can limit mail from remote processes within a specific classification range by setting up a limited range for the account.

The easiest procedure for setting up a specific account for the MAIL object is to use the SYS$MANAGER:NETCONFIG.COM command procedure, as described in Section 8.5 of the *Guide to VMS System Security*. To set up such an account so that mail will be allowed from any remote process to any local mail directory whose classification dominates the classification of the remote process, do the following:

1 Use SYS$MANAGER:NETCONFIG.COM to set up a specific account for the MAIL object. To do this answer YES to the question *"Do you want a default account for the MAIL object?"*. This will cause an account named MAIL$SERVER to be created.

2 Modify MAIL$SERVER to give it DOWNGRADE privilege as both an authorized and a default privilege.

3 Change MAIL$SERVER's classification to a full range.

4 Set MAIL$SERVER's default directory to an unclassified directory on a disk with a minimum classification of "(LEVEL:0, CATEGORY:NONE)".

If you have set up the mail object so that incoming proxy is enabled (see the *Guide to VMS System Security*, Section 8.6.1.3), the restriction described above relating to the DOWNGRADE privilege and default directory classification will apply to any proxy account used for mail.

Note: The mail object uses its default directory to buffer mail when receiving from the remote node. While it always deletes any file it creates for such

buffering, the system security officer should be aware that if mail is set up as described above, classified data might be written on an otherwise unclassified device. If this is a concern, ERASE ON DELETE can be enabled on the disk, but this may have an adverse on effect performance.

5.12 DECwindows in the Classified Environment

DECwindows usage is allowed in the classified environment. However, for this release of SEVMS, there are some restrictions which apply when logging in to a DECwindows session:

- You cannot specify a classification when logging in to a DECwindows session on a VAXstation. Instead, you must always log in at your *default* classification.
- For this version of SEVMS, you **must** log in to an *unclassified* DECwindows session (i.e. LEVEL=0, CATEGORIES=NONE). Attempting to log in to a classified DECwindows session will cause the following error message to be displayed:

```
UNSUPPORTED OPERATION OR FUNCTION
```

6 Auditing

This chapter describes the SEVMS-specific security auditing features of VMS. Complete information about security auditing can be found in the *VMS Audit Analysis Utility Manual* and the *Guide to VMS System Security* manual, which are a part of the VMS documentation set.

This chapter contains information about the following topics:

- SEVMS auditing commands and qualifiers
- SEVMS-specific portions of the binary audit record format

6.1 Introduction

SEVMS provides security auditing capabilities which are *in addition to* the standard capabilities provided by VMS. The additional features of SEVMS auditing are intended to support auditing of events related to mandatory access controls. Therefore, the use of SEVMS auditing does not preclude the use of VMS auditing; when using SEVMS auditing, you still need to employ standard VMS auditing features and practices.

To use SEVMS auditing, you should rely upon the *VMS Audit Analysis Utility Manual* and the *Guide to VMS System Security* for information about how to set up and use your system's standard VMS auditing features. This chapter is intended to provide you with information about any differences which SEVMS introduces to standard VMS auditing, and how to set up and use the SEVMS-specific features of auditing.

SEVMS auditing adds the following features to standard VMS auditing:

- Additional SEVMS-specific DCL commands and qualifiers pertaining to auditing
- The ability to selectively audit file and global section access by use of DOWNGRADE or UPGRADE privilege
- The ability to selectively audit file access by classification of the files
- The ability to selectively audit printing of files by classification of the files
- Recording the classification of the file for file-access security alarm messages
- Ability to audit $CHANGE_CLASS system service by classification of object

The following sections describe these capabilities in detail.

6.2 SEVMS Auditing Commands and Qualifiers

This section briefly describes the DCL auditing commands and qualifiers which SEVMS provides in addition to the standard VMS auditing commands. For a complete description of SEVMS commands and qualifiers, refer to the "DCL Commands" section of the *VMS SES User's Guide*.

For a description of VMS auditing commands and qualifiers, and information about how these commands are used, refer to the *VMS DCL Dictionary*, the *Guide to VMS System Security*, and the *VMS Audit Analysis Utility Manual*.

SEVMS adds the following things to VMS commands and qualifiers:

- SEVMS qualifiers are provided for the SET AUDIT command to enable auditing of events related to mandatory access control.
- SEVMS qualifiers are provided for the SHOW AUDIT command to display the mandatory access control auditing characteristics which are enabled.
- SEVMS keywords are provided for the ANALYZE AUDIT command.

The following sections of this chapter discuss the items in the above list in more detail.

6.2.1 SET AUDIT

The SET AUDIT command enables and disables the auditing of various types of events. SEVMS adds some new file access keywords and a new qualifier (/SECRECY) to the standard VMS command.

6.2.1.1 SET AUDIT/ALARM

SEVMS adds two new keywords, DOWNGRADE and UPGRADE, which can be used with the /ENABLE=FILE_ACCESS and /DISABLE=FILE_ACCESS qualifiers of the SET AUDIT/ALARM command. The new keywords, and the events for which they enable or disable alarms, are shown in Table 6-1.

Table 6-1 New Keywords For SET AUDIT/ALARM

Keyword:	Enables/Disables Alarms For:
DOWNGRADE[:access[,access]]	Successful file access due to the use of the DOWNGRADE privilege.
UPGRADE[:access[,access]]	Successful file access due to the use of the UPGRADE privilege.

Both of these keywords permit you to define the *type of file access* that is to be audited. See the SET AUDIT command in the *VMS DCL Dictionary* for a description of the SET AUDIT/ALARM command and *type of file access* keywords. For further details on how to use the *SET AUDIT* command, refer to the *VMS DCL Dictionary* and the *VMS SES User's Guide*.

Note: To display the status of these alarms, use the SHOW AUDIT command, not the SHOW AUDIT/SECRECY command described in Section 6.2.3.

6.2.1.2 SET AUDIT/SECRECY

The SET AUDIT/SECRECY command enables or disables security auditing of events related to the secrecy classifications enforced by SEVMS. This command is designed to be used in conjunction with the SET AUDIT/ALARM command, not to replace it. The result of using this command can be displayed with the SHOW AUDIT/SECRECY command.

Note: SEVMS also provides a SET AUDIT/INTEGRITY command; it operates in the same way as SET AUDIT/SECRECY, but relates to integrity classifications.

The SET AUDIT/SECRECY command causes alarms to be generated in the same form as the SET AUDIT/ALARM command. SET AUDIT/SECRECY controls only the auditing of events that are unique to SEVMS; it does not effect events controlled by the SET AUDIT/ALARM command.

The events controlled by the SET AUDIT/SECRECY command are:

- Successful access of files (based upon their classification)
- Unsuccessful access of files (based upon their classification)
- Files printed successfully by the SEVMS secure print symbiont (based upon their classification)
- Attempts to print files (using the SEVMS secure print symbiont) which fail because the classification of the file is not equal to, or within, the classification range of the printer
- Bypasses of printed output labeling (page headers and trailers) caused by the SEVMS secure print symbiont (by using the /PASSALL qualifier of the print command)
- Unsuccessful attempts to bypass printed output labeling by use of the /PASSALL qualifier
- Successful changes of object classification by means of the $CHANGE_CLASS system service (or SET CLASS DCL command, which uses that system service)
- Unsuccessful attempts to change object classification by means of the $CHANGE_CLASS system service (or SET CLASS DCL command, which uses that system service)

Each of these events can be enabled or disabled for specific classification levels and categories. If an audit is enabled for a given category, any object whose classification contains that category will be audited; its entire set of categories need not match the set of categories being audited. In the case of a successful change of object classification, the old object classification is used as the basis of auditing.

Both VMS and SEVMS audit only the *access* of files, not operations performed once the file has been accessed. In particular, changes in a file's attributes *including its classification* are not audited as file accesses. For example, assume a file called SWEETHEARTS.DAT has a TOP_SECRET

classification. In this example, suppose that a TOP_SECRET process successfully issues the following command:

```
$ SET CLASS/OBJECT_TYPE=FILE SWEETHEARTS.DAT/SECRECY=(LEVEL:0)
```

In this example, the use of DOWNGRADE or BYPASS privilege is *not* audited; a successful file access will be audited.

The SET AUDIT/SECRECY command permits you to monitor failed accesses of files with specific secrecy classifications. This technique is an excellent method for catching people who might probe into a specific project.

The SET AUDIT/SECRECY command is fully described in the *VMS SES User's Guide*.

Refer to the *Guide to VMS System Security* for further information about auditing with security alarms.

6.2.2 SET AUDIT/INTEGRITY

The SET AUDIT/INTEGRITY command enables or disables security auditing of events related to the integrity classifications enforced by SEVMS. This command works in the same manner as SET AUDIT/SECRECY, with the exception that it deals with integrity rather than secrecy.

The result of the SET AUDIT/INTEGRITY command can be displayed with the SHOW AUDIT/SECRECY command.

Note: **The /ALARM, /SECRECY and /INTEGRITY qualifiers are incompatible. They cannot be used in the same SET AUDIT command.**

The SET AUDIT/INTEGRITY command is fully described in the *VMS SES User's Guide*.

6.2.3 SHOW AUDIT

This command requires the use of the /SECRECY qualifier for SEVMS. The SHOW AUDIT/SECRECY command provides a display that identifies which security auditing features have been enabled with the SET AUDIT/SECRECY and SET AUDIT/INTEGRITY commands and the events that will be audited. This command is described in the *VMS SES User's Guide*.

This command is useful for checking which auditing features are enabled whenever you plan to add or delete features with a SET AUDIT/SECRECY or SET AUDIT/INTEGRITY command.

6.2.4 ANALYZE/AUDIT

The ANALYZE/AUDIT command analyzes the audit records of an audit archive file.

This command is the the same for both VMS and SEVMS, with the exception that SEVMS adds some extra keywords and criteria to a few of the qualifiers. Also, when relevant, SEVMS-related information is displayed in addition to standard VMS information. Other than that, both SEVMS and VMS use the ANALYZE/AUDIT command in the same way, and both use the same command qualifiers. Therefore, only SEVMS-specific usage of this command is explained in this section. For a full description of how to use ANALYZE/AUDIT, refer to the *VMS Audit Analysis Utility Manual* and the *Guide to VMS System Security*. For details about the SEVMS-specific features of ANALYZE/AUDIT, refer to the "DCL Commands" section of the *VMS SES User's Guide*.

SEVMS Keywords and Criteria

For SEVMS, some keywords and criteria have been added to VMS ANALYZE/AUDIT command qualifiers to enable analysis of mandatory access control events. The /SELECT qualifier can use the SEVMS-specific keywords UPGRADE and DOWNGRADE, and the SEVMS-specific SYMBIONT_ID and JOBNUM criteria.

The SEVMS-specific additions to ANALYZE/AUDIT are summarized in the following tables.

Additions to /SELECT Qualifier

The /SELECT qualifier specifies the criteria to be used when selecting event records. SEVMS adds SYMBIONT_ID and JOBNUM criteria, and PRIVILEGES_USED keywords to it.

Keyword	Description
PRIVILEGES_USED = (privs[,...])	Specifies the privileges of the process to be used in selecting event records. The following SEVMS privileges can be specified: UPGRADE DOWNGRADE
SYMBIONT_ID	Specifies the process ID for the SEVMS print symbiont.
JOBNUM	Specifies the number of the print job.

SEVMS Information Displayed With /SUMMARY Qualifier

When the ANALYZE/AUDIT/SUMMARY command is issued, in addition to the standard VMS information which is usually displayed, SEVMS adds information about "Files Printed" and "Classification Changes".

The output which results when this command is issued is shown in the following example. Note the SEVMS-related information which is included in this output.

```
$ AUDIT/ANALYZE/SUMMARY FOO.LOG

Total records read:     54          Records selected:        54
Record buffer size:     512         Format buffer size:      256
Server messages:        0           Customer messages:       0
Digital CSS messages:   0           Layered prod messages:   0
Audit changes:          7           Installed db changes:    0
Login failures:         0           Breakin attempts:        0
Successful logins:      3           Successful logouts:      0
System UAF changes:     0           Network UAF changes:     0
Rights db changes:      0           Object accesses:         22
File Printed:           4           Classification changes:  18
Volume (dis)mounts:     0
```

6.3 Alarm Messages

This section describes the SEVMS alarm messages which result from auditing system events related to mandatory access controls. These alarm messages are provided in addition to the standard VMS alarm messages that result from auditing various VMS system events.

The SEVMS alarms described in this section are: print symbiont alarms, label bypass alarms, and file access alarms.

For complete information about VMS alarm messages, refer to "Appendix E - Alarm Messages", in the *Guide to VMS System Security*.

6.3.1 Print Symbiont Alarms

You can audit print symbiont activity by specifying the PRINTED_FILE, and/or LABEL_BYPASS keywords with the /ENABLE *and* /SECRECY qualifiers of the SET AUDIT command. In addition to the information contained in all alarm messages, this type of alarm contains the following:

- Classification
- Symbiont Process ID
- File Name
- Device Name
- Job Number

The following messages are examples of alarms generated by the print symbiont.

Printed file success alarm

```
Security alarm (SECURITY) and security audit (SECURITY) on TURBO, system id: 2601
Auditable event:        Successfully printed file
Event time:             26-SEP-1989 07:40:27.01
Username:               OCONNELL
Object name:            _$255$DUA2:[OCONNELL]FOO.BAR;1
Object type:            file
Object min. class.:     SECRECY=(LEVEL=0,CATEGORY=(NONE))
Device name:            _TURBOSTXA4:
Symbiont PID            00000150
Job Number              920
```

Printed file failure alarm

```
Security alarm (SECURITY) and security audit (SECURITY) on TURBO, system id: 2601
Auditable event:        Protection check failed printing file
Event time:             26-SEP-1989 07:44:01.40
Username:               OCONNELL
Object name:            _$255$DUA2:[SECRET]LOGIN.COM;1
Object type:            file
Object min. class.:     SECRECY=(LEVEL=10,CATEGORY=(NONE))
Status:                 %SYSTEM-F-NOPRIV, no privilege for attempted operation
Device name:            _TURBO$TXA4:
Symbiont PID            00000150
Job Number              921
```

Label Bypass Successful Alarm

```
Security alarm (SECURITY) and security audit (SECURITY) on TURBO, system id: 2601
Auditable event:        Page header/trailer bypassed
Event time:             26-SEP-1989 07:40:27.01
Username:               OCONNELL
Object name:            _$255$DUA2:[OCONNELL]FOO.BAR;1
Object type:            file
Object min. class.:     SECRECY=(LEVEL=0,CATEGORY=(NONE))
Device name:            _TURBO$TXA4:
Symbiont PID            00000150
Job Number              920
```

Label Bypass Failure Alarm

```
Security alarm (SECURITY) and security audit (SECURITY) on TURBO, system id: 2601
Auditable event:        Page header/trailer bypassed failure
Event time:             26-SEP-1989 07:40:27.01
Username:               OCONNELL
Object name:            _$255$DUA2:[OCONNELL]FOO.BAR;1
Object type:            file
Object min. class.:     SECRECY=(LEVEL=0,CATEGORY=(NONE))
Device name:            _TURBO$TXA4:
Symbiont PID            00000150
Job Number              920
```

6.3.2 File Access Alarms

The following are examples of file access failure and file access success alarms.

File Access Failure Alarm

```
Security alarm (SECURITY) and security audit (SECURITY) on TURBO, system id: 2601
Auditable event:        Attempted file access
Event time:             26-SEP-1989 08:14:45.80
PID:                    00000198
Username:               OCONNELL
Image name:             $255$DUA3:[SYS0.SYSCOMMON.][SYSEXE]TYPE.EXE
Object name:            _$255$DUA2:[000000]SECRET.DIR;1
Object type:            file
Object min. class.:     SECRECY=(LEVEL=10,CATEGORY=(NONE))
Access requested:       EXECUTE
Status:                 %SYSTEM-F-NOPRIV, no privilege for attempted operation
```

File Access Success Alarm

```
Security alarm (SECURITY) and security audit (SECURITY) on TURBO, system id: 2601
Auditable event:        Attempted file access
Event time:             26-SEP-1989 08:16:26.44
PID:                    00000198
Username:               OCONNELL
Image name:             $255$DUA3:[SYS0.SYSCOMMON.][SYSEXE]TYPE.EXE
Object name:            _$255$DUA2:[SECRET]LOGIN.COM;1
Object type:            file
Object min. class.:     SECRECY=(LEVEL=10,CATEGORY=(NONE))
Access requested:       READ
Status:                 %SYSTEM-S-NORMAL, normal successful completion
Privileges used:        BYPASS
```

6.3.3 Classification Change Alarms

Classification Change Success Alarm

Successful change of object classification by means of the $CHANGE_CLASS system service (or SET CLASS DCL command which uses that system service).

```
Security alarm (SECURITY) and security audit (SECURITY) on TURBO, system id: 2601
Auditable event:        Successful classification change
Event time:             26-SEP-1989 08:16:26.44
PID:                    00000198
Username:               OCONNELL
Image name:             $255$DUA3:[SYS0.SYSCOMMON.][SYSEXE]TYPE.EXE
Object name:            _$255$DUA2:[SECRET]LOGIN.COM;1
Object type:            file
Object min. class.:     SECRECY=(LEVEL=10,CATEGORY=(NONE))
New Object min. class.: SECRECY=(LEVEL=0,CATEGORY=(NONE))
Access requested:       READ
```

Classification Change Failure Alarm

Unsuccessful attempt to change object classification by means of the $CHANGE_CLASS system service (or SET CLASS DCL command which uses that system service).

```
Security alarm (SECURITY) and security audit (SECURITY) on TURBO, system id: 2601
Auditable event:        Attempted classification change
Event time:             26-SEP-1989 08:16:26.44
PID:                    00000198
Username:               OCONNELL
Image name:             $255$DUA3:[SYS0.SYSCOMMON.][SYSEXE]TYPE.EXE
Object name:            _$255$DUA2:[SECRET]LOGIN.COM;1
Object type:            file
Object min. class.:     SECRECY=(LEVEL=10,CATEGORY=(NONE))
New Object min. class.: SECRECY=(LEVEL=0,CATEGORY=(NONE))
New Object max. class.: SECRECY=(LEVEL=0,CATEGORY=(NONE))
Status:                 %SYSTEM-F-NOPRIV, no privilege for attempted operation
```

6.3.4 Binary Audit Record Format

SEVMS uses a binary audit record format which is the same as the standard VMS binary audit record format, with some SEVMS-specific additions.

As in VMS, the SEVMS binary audit record format consists of a header packet followed by a series of individual data packets. The header packet contains information about the type of event being audited and the number of data packets which follow. The data packets contain the actual information about the audited event. SEVMS adds several record and packet types related to mandatory access controls to the standard information provided by the VMS audit record.

For complete information about the internal structure and contents of the standard VMS binary audit record, refer to "Appendix A - Security Audit Message Format", in the *VMS Audit Analysis Utility Manual*.

The "Programming Information" section of the *VMS SES User's Guide* describes those SEVMS-specific portions of the binary audit record.

7 The SEVMS Secure Print Facility

This chapter describes the SEVMS secure print facility. The secure print facility of SEVMS provides labeling of printed output. It is implemented with two print symbiont images - SEVMS_SMB.EXE and SEVMS$LATSYM.EXE.

SEVMS_SMB.EXE provides support for printers directly connected to a system. SEVMS$LATSYM.EXE provides support for printers connected to a system by means of LAT terminal servers.

Both SEVMS print symbionts operate, and are used, in an identical manner (with the exception of supporting LAT printers). Neither SEVMS print symbiont provides support for POSTSCRIPT printers.

The following topics are discussed in this chapter:

- Logical names for the print symbiont
- Using print symbiont templates
- Invoking the print symbiont
- Printing listing files
- Graphic printer considerations

7.1 Introduction

The SEVMS print symbionts are alternatives to the VMS standard print symbiont and VMS standard LAT print symbiont. To use either of the SEVMS print symbionts, you must specify them when initializing a print queue. Any queues initialized without specifying an SEVMS print symbiont will use the VMS standard print symbiont. Refer to Section 7.5 for information about how to start an SEVMS print symbiont.

The SEVMS print symbionts are designed to control and format classified printed output, where such information must be clearly labeled. It is NOT a customizable print symbiont, as is the VMS print symbiont. However, considerable latitude is available for customizing the format of a label and the information it contains.

Labels can include the following:

- Date of printing
- Name of the file being printed
- User name of the person printing the document
- Document expiration date
- Page number
- Total number of pages in the file

- Total number of pages actually printed
- Job number
- Device on which the file was printed
- Queue on which the print job was submitted
- Random number (to be associated with the front page/back page pair)
- File's secrecy classification
- Lines of text produced by customer-supplied formatting routines

Aside from labeling the output, the SEVMS print symbionts check the classification of each file it prints and ensure that no file is printed if its classification is outside of the classification range of the printer.

The print symbionts can also generate security alarms when those alarms have been enabled with the SET AUDIT/SECRECY command. See Chapter 6 for more information about this topic.

7.2 Print Symbiont Logical Names

In this chapter, several system-wide logical names are described that can be used to control SEVMS print symbiont operation. These logical names have the form *LOGICAL-NAME_ddcu*. In these logical names, the string *ddcu* represents the actual device specification of some printer device. Thus, for example, the logical name SEVMS$SMB_FILE_FRONTPAGE_LPA0 could be defined to affect the operation of the printer device LPA0.

On systems where the SYSGEN parameter SCSNODE is defined, the device specification of a printer device has the form *SCSNODE$ddcu* (e.g. MYNODE$LPA0). Such systems include, among others, those that are part of a VAXcluster system. On such a system, a print symbiont logical name may be defined using either the form *LOGICAL-NAME_ddcu*, or the form *LOGICAL-NAME_SCSNODE$ddcu*. Either form will be recognized by the SEVMS print symbiont, which looks for the *LOGICAL-NAME_SCSNODE$ddcu* form first.

Note: When specifying the printer device name, do not include a colon (:) after the device name.

7.3 Print Symbiont Templates

The format of a label is defined in a *template*. There can be many different templates.

The following things determine which template a print symbiont uses to label a file:

- Secrecy categories
- Secrecy level
- Printer
- Printer width

This is useful for meeting various output labeling requirements.

If there is no specific template associated with the characteristics of a print job, a default template is used. SEVMS provides a predefined default template which can be modified.

7.3.1 Selecting a Template

This section describes the way in which a print symbiont selects and locates a template for a print job.

7.3.1.1 Overview

The user who prints a file has no direct control over which particular template is used to print that file.

Instead, a print symbiont selects a template for a print job based upon the following set of criteria: file classification, printer, and printer width.

A print symbiont uses the above criteria as a key to locate a record in an RMS ISAM file ("SYS$LIBRARY:SEVMS$SMB_HDRFRM.DAT"). The record, if found in this file, specifies the template to be used.

If a print symbiont cannot find a template based upon the above criteria, then a default template is used (either the NO_CLASS template or the SEVMS_DEFAULT template).

Details about the method in which a print symbiont selects a template are provided in the following portions of this section.

7.3.1.2 Template Selection Process

A print symbiont bases the selection of a template for a print job upon the following criteria:

- File classification (secrecy categories, secrecy level)
- Printer
- Printer width

A print symbiont uses these criteria as a key to locate a record in an RMS ISAM file. The records in this file specify defined templates to be used. Each record contains one or more of the items in the above list, and the name of a template. This is referred to as an *association*. The file is created and maintained by the SET TEMPLATE command.

The RMS ISAM file can be indicated by the logical name "SEVMS$SMB_HDRFRM" (but is assumed to be "SYS$LIBRARY:SEVMS$SMB_HDRFRM.DAT" if the logical name is not defined). In other words, the SYS$LIBRARY:SEVMS$SMB_HDRFRM.DAT file simply correlates defined critera with defined templates.

To select a template, a print symbiont follows a predefined process. The symbiont first looks in the database for a template with an association that matches the printer, and the file's secrecy classification, exactly with a width less than or equal to the width of the printer. If there is no such template, the symbiont tries for a match with a template associated with an *incomplete* set of characteristics.

The order matches are checked for is:

1 Printer, Categories, Level, Width

2 Printer, Categories, Width

3 Printer, Level, Width

4 Printer, Width

5 Categories, Level, Width

6 Categories, Width

7 Level, Width

8 Width

If a print symbiont cannot find a template to match the specified criteria (i.e. if no record is found), then either the NO_CLASS template or the SEVMS_DEFAULT template is used.

If the security manager has defined the template named "NO_CLASS", then the symbiont uses that template if the file to be printed has no classification (i.e. secrecy level = 0, secrecy categories = none).

If the "NO_CLASS" template has not been defined, or, if the file's classification is other than zero, then the "SEVMS_DEFAULT" template is used as the default template by the symbiont. This allows the security manager the option to associate a distinct template with files which cannot be matched to the above criteria.

NOTE: The NO_CLASS template is supported to provide compatibility with earlier versions of SEVMS.

The default template, "SEVMS_DEFAULT", is provided with the SEVMS kit. This template can also serve as a guide for building additional templates. The "SEVMS_DEFAULT" template is shown in *Appendix A* of this manual.

Having determined the name of the template to use, a print symbiont then retrieves it from a text library that can be indicated by the logical name "SEVMS$SMB_LIB". If the logical name is not defined, the file is assumed to be "SYS$LIBRARY:SEVMS$SMB_LIB.TLB".

7.3.2 Establishing and Using Templates

This section describes how to go about setting up templates and making them available to your system's print symbiont.

There are several steps required to establish a template for a print symbiont.

1 Determine the template(s) needed.

2 Design the template file, using an ordinary editor.

3 Store the template file, placing it in the symbiont template text library with the librarian. (For example, SYS$LIBRARY:SEVMS$SMB_LIB.TLB.) See the *VMS Librarian Utility Manual* for instructions on use of the LIBRARY Utility.

4 Associate the template file with a set of characteristics, using the SEVMS "SET TEMPLATE" command.

These steps are fully described in the following sections.

7.3.2.1 Step 1 - Determining Templates Needed

The specific labeling requirements at each site determine how many different templates will be needed and how they must be associated. The labeling requirements may vary by specific compartments (i.e. categories), or by clearances (i.e. levels), or by a combination of both.

SEVMS provides four different label elements:

- Front pages
- Back pages
- Page headers
- Page trailers

These label elements are described in detail in Section 7.3.2.2.1. However, because the way a print symbiont deals with front and back pages can affect the implementation of labeling requirements, it is important to know about them before determining needed templates. This information is explained in the next section.

7.3.2.1.1 Front and Back Page Considerations

The SEVMS print symbiont allows the user to specify the template for both a front page, which precedes the first page of a file or job, and a back page, which follows the last page of a file or job. Whenever the decision is made to print a front page or a back page, such page is printed twice.

The procedure used to format front and back pages is described in the following sections. The decision as to whether to surround a given file with front and back pages is based on the criteria provided in this section.

In order for a front page and back page to be printed, the template associated with the current print file must contain the .FRONTPAGE and .BACKPAGE directives, with associated text describing the layout of each of these pages. In this case, two front pages are ALWAYS printed before the first file of a job, and two corresponding back pages are ALWAYS printed after the last file of that job. (If a job has only one file in it, then that file is surrounded by two front and back pages.) If a job has several files in it, and if the classification of any of those files differs from the classification of the file immediately preceding it, then two back pages are printed after the preceding file, and two new front pages are printed before the file whose classification differs.

Note: There is one situation in which the printing of an SEVMS back page can be disabled. If a STOP/QUEUE/RESET command is executed while a file is being printed, the print job is aborted immediately and no back pages are printed. The STOP/QUEUE/RESET command is restricted to users with the OPER privilege or EXECUTE access to the queue.

In addition, there is a set of system-wide logical names that can be defined to further control the printing of front and back pages. These logical names have the format SEVMS$SMB_FILE_FRONTPAGE_ddcu. (*ddcu* is the device specification of the target printer.)

If the logical name SEVMS$SMB_FILE_FRONTPAGE_ddcu is undefined, or is defined to be anything other than "1", the criteria described above is used to decide when to print front and back pages. However, if SEVMS$SMB_FILE_FRONTPAGE_ddcu is defined to be equal to "1", then, for that particular print device, each and every file is preceded by two front pages and followed by two back pages.

Note that the logical names described above are only checked when a print queue is initialized. Defining, or redefining these logical names has no effect on already existing queues. Thus, to use these logical names properly, make sure they are set to the desired values before initializing a given print queue.

7.3.2.1.2 Graphics Printer Considerations

For graphics printers, a user can specify "/PASSALL" in the PRINT command to avoid the carriage control formatting that is normally done by the symbiont. However, use of the /PASSALL switch also disables the output of the SEVMS page headers and page trailers. Since disabling page headers and page trailers has a potential security impact, there are some constraints on the use of the /PASSALL switch. These constraints are controlled by the following system-wide (DEFINE/SYSTEM) logical names: PSM$PASSALL_NOBYPASS and SEVMS$SMB_LABEL_BYPASS_ddcu.

In the above logical name, *ddcu* is the device specification of the target printer.

If PSM$PASSALL_NOBYPASS is defined to be "1", then any use of the /PASSALL switch in a print command is ignored, and the file is printed as if /PASSALL had not been specified.

If PSM$PASSALL_NOBYPASS is undefined, or is defined to be anything other than "1", then the logical name, SEVMS$SMB_LABEL_BYPASS_ddcu, becomes important. In this situation, if SEVMS$SMB_LABEL_BYPASS_ddcu is defined to be "1", then the /PASSALL switch takes effect and no page headers or page trailers are printed.

If neither PSM$PASSALL_NOBYPASS nor SEVMS$SMB_LABEL_BYPASS_ddcu are defined to be "1", then the use of the /PASSALL switch causes the job to abort, and an SEVMS_E_NOPASSALL error is displayed on the printer.

These rules are summarized in the following table:

NOBYPASS	LABEL_BYPASS	/PASSALL
1	ignored	ignored
X	1	standard symbiont
X	X	abort job

Both bypassing of page headers/trailers, and aborting due to a disallowed /PASSALL switch, can be audited. Refer to Chapter 6 of this manual and the *VMS SES User's Guide* for more information about auditing these events

Note that the logical names described above are only checked when a print queue is initialized. Defining, or redefining, these logical names has no effect on already existing queues. To use these logical names properly, make sure they are set to the desired values before initializing a given print queue.

For example, consider the following situation. A site has 3 print queues: LP, LN03, and LQP, each one associated with a separate printer. On LP, any use of the /PASSALL switch is to be ignored and not audited. On LN03, the use of the /PASSALL switch is permitted, but the ability to audit such use is desired. On LQP, any attempt to use the /PASSALL switch should cause the associated job to abort.

The following command sequence can be used to support this configuration:

```
$ DEFINE/SYSTEM PSM$PASSALL_NOBYPASS 1
$ INITIALIZE/QUEUE/START/PROCESSOR=SEVMS_SMB/ON=LPA0: LP

$ DEFINE/SYSTEM PSM$PASSALL_NOBYPASS 0
$ DEFINE/SYSTEM SEVMS$SMB_LABEL_BYPASS_TXA2 1
$ INITIALIZE/QUEUE/START/PROCESSOR=SEVMS_SMB/ON=TXA2: LN03

$ DEFINE/SYSTEM SEVMS$SMB_LABEL_BYPASS_TXA7 0
$ INITIALIZE/QUEUE/START/PROCESSOR=SEVMS_SMB/ON=TXA7: LQP
```

7.3.2.2 Step 2 - Designing Templates

Templates are text files that can be created with any VMS text editor. They can contain literal text which is printed "as is" on the page, as well as directives that are interpreted by a print symbiont. There are three kinds of directives:

- *Element directives* which delimit the different elements of the template.
- *Field directives* which represent information that is different for each printed file, such as the file's name, or changes as the file prints, such as the page number.
- *Format directives* which control the way the literal text and field directives are printed (for instance - as large block letters).

More information on the directives and how they are used appears in the following sections.

After a template file has been edited, it must be stored in a print symbiont's template library. Finally, the DCL command SET TEMPLATE must be used to associate the template with one or more print job characteristics. More information on this topic process is provided in the following sections.

7.3.2.2.1 Template Elements

Each template consists of four optional elements, delimited by element directives. Each element can contain any combination of literal text, field directives and format directives.

The four elements are:

- A *front page* which is printed based upon the criteria described in Section 7.3.2.1.1.
- A *back page* which is printed based upon the criteria described in Section 7.3.2.1.1.
- A *page header* which is printed at the top of each page of output, unless the file was printed /PASSALL. (See the section on "Graphic Printer Considerations", Section 7.3.2.1.2 for more information about /PASSALL.)
- A *page trailer* which is printed at the bottom of each page of output, unless the file was printed /PASSALL.

When an element is printed, each line of the element in the template, including blank lines, results in a line of output. If a line contains any field directives, they are expanded and the resulting line is printed. If a line contains only text, the line is printed without any changes.

The size of each element, *after* field and format directives are expanded, is limited by the width and page length of the printer(s) or form(s) that it is printed on. If the number of characters in the expanded element line exceeds the width or the number of lines in the element exceeds the page length, an error is generated and the job is aborted. Care should be taken when designing templates to ensure these limits will not be exceeded.

7.3.2.2.2 Element Directives

The element directives delimit the various elements in the template. The element itself begins on the line immediately following the directive and continues until the next element directive, or the end of the template file.

There are several rules that should be observed when using element directives:

1. An element directive should begin in the first column of a line.
2. An element directive should be the only thing on the line.
3. Each element directive should appear no more than once in each template file.
4. The size of each element, *after* field and format directives are expanded, is subject to the following limits:
   - The number of characters in each line of the element should not exceed the page width.
   - The size of any element, in characters, is limited to the page size (width * length).
   - The number of lines in the page header plus the number of lines in the page trailer must be *less than* the length of the page.

5 If a given template is to be used on several print pages of different sizes, the above limits should be considered relative to the smallest print pages.

6 The use of explicit vertical form control (the form feed character) is *not* recommended.

.FRONTPAGE

This directive allows the user to define a custom page which is printed based upon the criteria described in Section 7.3.2.1.1.

.BACKPAGE

This directive allows the user to define a custom page which is printed based upon the criteria described in Section 7.3.2.1.1.

.HEADER

This directive allows the user to define a custom page header which is printed at the top of each page of each file, but not on front, back, trailer or flag pages. The .DOUBLE and .TRIPLE format directives (described below) cannot be used in a page header.

.TRAILER

This directive allows the user to define a custom page trailer which is printed at the bottom of each page of each file but not front, back, trailer, or flag pages. The .DOUBLE and .TRIPLE format directives (described below) cannot be used in a page trailer.

7.3.2.2.3 Field Directives

A field directive indicates a text string, determined at the time a job is printed, that should be printed at the directive's location in the element. Some of these text strings are variable in length, others have a fixed length. The directives and their length are shown in Table 7-1, which follows.

Table 7-1 Lengths of Field Directives

Field Directive	Minimum Length	Maximum Length
.SEC_LEV	1	29
.SEC_CAT	1	3712
.DATE	17	17
.DELDATE	17	17
.FILENAME	3	71
.FORMAT_LINE	1	Not applicable
.PAGE	1	5
.TPAGE	1	5
.USERNAME	13	32
.JOBNUM	1	10
.DEVICE	4	15

Table 7-1 (Cont.) Lengths of Field Directives

Field Directive	Minimum Length	Maximum Length
.QUEUE	1	31
.RANDOM	1	10
.PPAGE	1	5

Field directives are expanded and placed in the output line using the following rules:

- The resultant string begins at the location in the line given by the "." of the directive.
- If the resultant string is shorter than the directive (as it appears in the template), it is padded with blanks to the length of the directive.
- If the resulting string is the same length as the directive, it replaces the directive.
- If the resultant string is longer than the directive, the first part of the string (equal to the length of the directive) replaces the directive. Then:
  - Any blanks following the directive are replaced with the remainder of the string.
  - If there are not enough blanks to contain the entire remainder, the rest of the output line (in respect to the template) is shifted right to allow room for the entire string plus one extra space.

For example, assume that the following is a template line:

```
******** .FILENAME ********
```

If the file's name is A.B;1, then the output would be:

```
******** A.B;1      ********
```

If the file's name is ABC.DEF;1, then the output would be:

```
******** ABC.DEF;1 ********
```

And if the file's name is ABCDEFGHIJKLMNOP.QRST;1, then the output would be:

```
******** ABCDEFGHIJKLMNOP.QRST;1 ********
```

.SEC_LEV

At print time, a print symbiont replaces each occurrence of this directive with the text representing the file's secrecy level.

.SEC_CAT

At print time, a print symbiont replaces each occurrence of this directive with the text representing the file's secrecy categories.

.DELDATE <number-of-days>

At print time, a print symbiont replaces each occurrence of this directive with a date and time which is "<number-of-days>" days after the date of printing. "<number-of-days>" must be a decimal integer and normally represents the number of days before the document is to be destroyed.

.DATE

At print time, a print symbiont replaces each occurrence of this directive with the current date and time.

.FILENAME

At print time, a print symbiont replaces each occurrence of this directive with the filename of the file it is printing.

.PAGE

At print time, a print symbiont replaces each occurrence of this directive with the current page number. The .PAGE directive can *not* be used on the front page of a print job. Also, the .PAGE directive is meaningless on a file that is printed with the /PASSALL switch, since no page formatting is done.

.TPAGE

At print time, a print symbiont replaces each occurrence of this directive with the total number of pages in the file. The .TPAGE directive can *not* be used on the front page of a print job. Also, the .TPAGE directive is meaningless on a file that is printed with the /PASSALL switch, since no page formatting is done.

Note: **The .TPAGE directive describes the total number of pages in the file, *NOT* the number of pages actually printed. These two numbers may differ. For instance, if the job is aborted before it completes, or if the user prints only some of a file's pages (via the PRINT/PAGE=(n,m) or the SET QUEUE/ENTRY=xxx/PAGE=(n,m) commands), or if the user pauses a job (using STOP/QUE) and then specifies a different page to resume printing (e.g. START/QUE/TOP.OF.FILE), then .TPAGE may be different than the number of pages actually printed.**

.PPAGE

At print time, a print symbiont replaces each occurrence of this directive with the number of pages that have actually been printed since the most recent front page. The .PPAGE directive can not be used on the front page of a print job. Also, the .PPAGE directive is meaningless on a file that is printed with the /PASSALL switch, since no page formatting is done.

Note: **If a job is aborted with the STOP/ENTRY=n or DELETE/ENTRY=n commands, the number of pages actually printed is not predictable. In this case, the .PPAGE directive, if used in a back page, would be replaced with the string "?????".**

.USERNAME

At print time, a print symbiont replaces each occurrence of this directive with the username of the person who queued the print request.

.JOBNUM

At print time, a print symbiont replaces each occurrence of this directive with the job number (entry number) assigned to the job by the queue manager.

.DEVICE

At print time, a print symbiont replaces each occurrence of this directive with the name of the device on which the job was printed.

.QUEUE

At print time, a print symbiont replaces each occurrence of this directive with the name of the queue on which the job was submitted. If a job is submitted to a generic queue, then *its* name replaces the .QUEUE directive. (Rather than the name of the device queue on which the job is ultimately queued.)

.RANDOM

At print time, a print symbiont replaces each occurrence of this directive with a random number that is associated with the current job (or file). Such a random number is generated whenever a front page is scheduled to be printed (even if the front page is blank). This random number remains the same until a new front page is scheduled. This can be used to associate an unpredictable number with a front page/back page pair to prevent the "spoofing" of such pages inside a text file.

.FORMAT_LINE

This directive gives a print symbiont the ability to interface with customer-supplied formatting routines, whose output is placed into the label of the file being printed. Each use of the .FORMAT_LINE directive causes a single line of customer-supplied text to appear in the current label. This directive is described in detail in Section 7.4, "Customer-Supplied Formatting Routines."

7.3.2.2.4 Format Directives

The two format directives, .DOUBLE and .TRIPLE, cause the text and expanded directives in the following line to be output in large block "letters". These "letters" are created by printing multiple line patterns that look like large letters. These are the same sort of letters that appear on VMS print symbiont flag pages.

The following things should be noted concerning the use of the .DOUBLE and .TRIPLE directives:

- .DOUBLE and .TRIPLE can only be used in back pages and front pages.
- If a line is too long to fit on a page printed in .TRIPLE mode, it will be printed in .DOUBLE mode.

- If a line is too long to fit on a page printed in .DOUBLE mode, it will be printed in normal (single line) mode.
- Some special characters may not be converted to block characters properly. They can still be identified, for even though the block character is distorted, the block is composed of many instances of the original character.
- The characters printed are centered horizontally on the page.

.DOUBLE

This directive causes the next line of the template to print as "double" height characters. Double height characters are 7 lines high and 6 characters wide.

.TRIPLE

This directive causes the next line of the template to print as "triple" height characters. Triple height characters are 14 lines high and 12 characters wide.

7.3.2.2.5 Directive Usage Notes

- **All directives must be capitalized.**
- Directives can be abbreviated as long as the abbreviation is unambiguous with respect to the other directives.
- Except were previously noted, the directives can be placed anywhere in the template file and can be repeated as many times as desired.
- When using any of the field directives, it is important to remember how large each field could be, since the file will not be printed if the expanded template line is longer than the width of the printer.

7.3.2.3 Step 3 - Storing Templates

Place the template file into the symbiont template text library with the librarian. (For example, SYS$LIBRARY:SEVMS$SMB_LIB.TLB.) See the *VMS Librarian Utility Manual for instructions on use of the LIBRARY Utility.*

The following example shows the two template definition text files referenced above being placed in the text library:

```
$ LIBRARIAN/TEXT/INSERT SYS$LIBRARY:SEVMS$SMB_LIB RED_TEMPLATE.TXT, -
_$ BLUE_AND_GREEN_TEMPLATE.TXT
```

Note: The symbiont must not be active when storing templates in the library.

7.3.2.4 Step 4 - Associating Templates

Once a collection of templates has been designed and stored in the text library, the system manager must make them available to the print symbiont(s) on the system. The manner in which this is done is described in this section.

7.3.2.4.1 Template Association

The SET TEMPLATE command is used to associate a template with one or more of the following criteria: a printer, a set of secrecy categories, a secrecy level, and a form width. A template can be associated with more than one set of characteristics, and incomplete sets can be specified.

Refer to Section 7.3.1.2 for an explanation of the criteria used by the print symbiont to make a template selection.

An example set of associations is shown in Table 7-2. The SET TEMPLATE command is described in Section 7.3.2.4.4 of this manual and in the "Command Description" section of the *VMS SES User's Guide*.

In Table 7-2, the template "GENERIC_CLASSIFIED" is associated with more than one set of characteristics. Only the templates SPECIAL_1 and SPECIAL_1_WIDE are associated with a complete set of characteristics. A specific set of characteristics can only be associated with one template; however, there are no restrictions on the combinations of characteristics that can be associated with a template, and no limit to the number of different associations for each template.

Table 7-2 Sample template associations

Template	Printer	Categories	Level	Width[1]
SPECIAL_1_WIDE	LPA0	BLUE,GREEN	SECRET	132
SPECIAL_1	LPA0	BLUE,GREEN	SECRET	80
ONLY_BLUE		BLUE		
ANY_SECRET			SECRET	
ONLY_SECRET		None[3]	SECRET	
GENERIC_CLASSIFIED			CONFIDENTIAL	
GENERIC_CLASSIFIED			INTERNAL_USE	
WIDE_TOP_SECRET			TOP_SECRET	132
LN03	LN03_PRINTER[2]			
UNCLASS_132			0	132

[1]A blank width implies 0.

[2]An *SEVMS logical printer* name.

[3]A blank indicates no entry; None indicates an entry of no categories.

7.3.2.4.2 SEVMS Logical Printers

A printer can be specified by either be a physical device name or an *SEVMS logical printer* name.

An *SEVMS logical printer* name allows a template to be associated with more than one, but not all, physical printers. It should be noted that two different *SEVMS logical printer* names cannot be assigned to one physical printer. In order to give a printer an *SEVMS logical printer* name, a logical name of the form *SEVMS$SMB_PRINTER_ddcu*, whose equivalence name is the *SEVMS logical printer* name, must be defined. (In this case, ddcu is the physical device name of the printer.)

Note the following example:

```
$ DEFINE/SYSTEM SEVMS$SMB_PRINTER_TTA7 LN03_PRINTER
$ DEFINE/SYSTEM SEVMS$SMB_PRINTER_TTB4 LN03_PRINTER
```

In this example, these commands assign two printers the same SEVMS logical printer name. The logical names must be defined on the system upon which the symbiont is running. SEVMS logical printer names can also be used to establish generic symbiont databases which do not refer to physical devices.

On a cluster where two nodes each have a printer device of the same name, the *LOGICAL-NAME_SCSNODE$ddcu* form of the logical name can be used to distinguish between the two printers.

7.3.2.4.3 Examples

The following examples show which template would be selected for the indicated files if the symbiont database contained the associations shown in Table 7-2.

Refer to Section 7.3.1.2, which describes the criteria used by the print symbiont to make a template selection, to better understand the these examples.

```
Level=0, Categories=NONE, Printer=LPA2, Form width=80 : Template=NO_CLASS
```

"UNCLASS_132" matches level "0" and categories "none", but the form width is less than "132". Therefore, "NO_CLASS" is used.

```
Level=0, Categories=RED, Printer=LPA2, Form width=80 : Template=SEVMS_DEFAULT
```

There are no matching associations. Therefore, "SEVMS_DEFAULT" is used.

```
Level=CONFIDENTIAL, Categories=RED, Printer=LPA2, Form width=80 : Template=GENERIC_CLASSIFIED
```

The first match is of the "level" only. Therefore, "SEVMS_DEFAULT" is used.

```
Level=SECRET, Categories=GREEN, Printer=LPA0, Form width = 80 : Template=SEVMS_DEFAULT
```

"SPECIAL_1" is not selected because its category mask is "BLUE" *and* "GREEN", not just "GREEN". Therefore, "SEVMS_DEFAULT" is used.

```
Level=SECRET, Categories=(BLUE,GREEN), Printer=LPA0, Form width=80 : Template=SPECIAL_1
```

An exact match. Therefore, "SPECIAL_1" is used.

```
Level=SECRET, Categories=(BLUE,GREEN), Printer=LPA0, Form width=79 : Template=SEVMS_DEFAULT
```

"SPECIAL_1" is not selected because its width is wider than the form's width. Therefore, "SEVMS_DEFAULT" is used.

7.3.2.4.4 Using the SET and SHOW TEMPLATE Commands

The SET TEMPLATE command is used to associate templates with category masks, secrecy levels, printers, and printer widths, or to remove such associations. The SHOW TEMPLATE command is used to display the current association of templates with category masks, secrecy levels, printers, and printer widths,

These commands **do not** maintain the symbiont text library (SYS$LIBRARY:SEVMS$SMB_LIB.TLB) in which the template definitions are actually stored; this is done using LIBRARIAN commands.

For information about using the SET TEMPLATE and SHOW TEMPLATE commands, refer to the "Command Descriptions" section of the *VMS SES User's Guide*.

7.4 Customer-Supplied Formatting Routines

This section describes how customer-supplied formatting routines are constructed and used with the SEVMS print symbiont.

The topics discussed in this section are listed below:

- Introduction to customer-supplied formatting routines
- Symbiont/formatting-routine interface
- Creating and using formatting routines
- Notes about use of customer-supplied formatting routines

7.4.1 Introduction

The SEVMS print symbiont has the capability to interface with customer-supplied formatting routines. This capability is provided by the use of the .FORMAT_LINE template directive. For each .FORMAT_LINE directive in a template, a print symbiont calls a customer-supplied routine with a series of arguments. The customer-supplied routine then returns a single line of text to the symbiont, which is placed in the label at the location corresponding to the .FORMAT_LINE directive. The customer-supplied routine must be in a shareable image residing in SYS$SHARE.

The format of the .FORMAT_LINE directive is as follows:

```
.FORMAT_LINE routine,image,user-arg
```

In the preceeding format example, *routine* is the name of the formatting routine entry point, *image* is the filename of the shareable image containing the formatting routine, and *user-arg* is a customer-supplied argument to be passed to the formatting routine. All arguments are text strings, whose valid characters are the following:

- Uppercase or lowercase letters A through Z
- Dollar sign ($)
- Underscore (_)

All arguments must be specified. A space must separate the *routine* argument from the .FORMAT_LINE directive, and a comma (,), *not a space*, must separate the arguments from each other. This is demonstrated in the previous format example. Note that the *image* argument contains the filename only; the directory specification, file type, and version number cannot be specified in this argument.

7.4.2 The Symbiont/Formatting-Routine Interface

The SEVMS print symbiont conforms to the *VAX Procedure Calling Standard*, which is described in the *Introduction to VMS System Routines* manual of the "VMS Programming Subkit". When a print symbiont encounters a .FORMAT_LINE template directive, it calls the specified formatting routine with the following arguments, in the following order:

- file-class
- file-spec
- user-name
- device-name
- form-width
- user-arg
- line-ptr

file-class—The classification of the file being printed. The *file-class* argument is the address of 20-byte classification block, whose format is given in the "Programming Information" chapter of the *VMS SES User's Guide*.

file-spec—The file specification of the file being printed. The *file-spec* argument is the address of a string descriptor, whose format is *filename.type;version*.

user-name—The name of the user who submitted the file for printing. The *user-name* argument is the address of a string descriptor.

device-name—The name of the device on which the file is being printed. The *device-name* argument is the address of a string descriptor.

form-width—The size in characters of one line of the print form. The *form-width* is the address of a longword containing this size.

user-arg—The user argument that was specified in the .FORMAT_LINE directive. The *user-arg* argument is the address of a string descriptor.

line-ptr—Specifies where the formatting routine is to return the formatted line to the print symbiont. The formatting routine returns in *line-ptr* the address of a string descriptor for the formatted line. The *line-ptr* argument is the address of a longword.

7.4.3 Creating and Using Formatting Routines

To incorporate a customer-supplied formatting routine, use the following steps:

1. Edit and compile a routine which accepts the arguments passed by the symbiont; use them to return to the symbiont, by way of the *line-ptr* argument, a string descriptor for a line of text to be printed. The formatting routine may be written in any language which supports the *VAX Procedure Calling Standard*. The routine entry point must be a global symbol.
2. Link the formatting routine into a shareable image as described in the *VMS Linker Utility Manual*; use the "UNIVERSAL=" linker option to direct the linker to make the routine entry point a universal symbol. The shareable image must have a file type of ".EXE".
3. Place the shareable image into SYS$SHARE.
4. Place one or more instances of the .FORMAT_LINE directive into a new or existing template; specify the routine name, the shareable image name, and an argument to be passed to the routine.
5. Make the template available to the SEVMS print symbiont in the manner described in Section 7.3.2.

To aid in the development and debugging of customer-supplied formatting routines, SEVMS provides a test module, SEVMS$SMB_TEST.MAR, which resides in SYS$EXAMPLES. Instructions on the use of this test module are included in its source code.

7.4.4 Usage Notes

Several different customer-supplied formatting routines can exist in one or more shareable images.

In addition to the string descriptor for the formatted line returned by way of the argument list, the formatting routine must return a condition code as a function value. If the condition code has a severity of WARNING, ERROR, or SEVERE, the print job will terminate and a message corresponding to the condition value will be displayed on the printer. Condition codes are described in the *Introduction to VMS System Routines* manual of the "VMS Programming Subkit".

To enlarge the line returned by the formatting routine, a .TRIPLE or .DOUBLE template directive can be used on the line immediately preceding a .FORMAT_LINE directive. However, the restrictions stated in Section 7.3.2.2.4 should be noted.

Caution: A customer-supplied formatting routine runs in the context of the SEVMS print symbiont. Because of this, programming errors in such a routine could cause the SEVMS print symbiont image to abort. If this occurs, any queues that use the SEVMS print symbiont must be reinitialized and restarted. Therefore, take great care to ensure the integrity of a customer-supplied formatting routine before using it on your system.

The formatting routine must also comply with the restrictions concerning user-written routines which are discussed in VMS documentation. Refer to Section 10.3.1 of the VMS *Utility Routines* manual located in the VMS *System Routines* volume for this information.

7.5 Invoking A Print Symbiont

The SEVMS print symbionts are alternatives to the VMS standard print symbionts. The SEVMS print symbiont is called "SEVMS_SMB.EXE"; the VMS standard print symbiont is called "PRTSMB.EXE". The SEVMS LAT print symbiont is called "SEVMS$LATSYM.EXE"; the VMS LAT print symbiont is called "LATSYM.EXE".

The SEVMS print symbionts must be specified when starting individual print queues. The "/PROCESSOR=" qualifier is used to do this. An example of this is shown in the following commands:

```
$ INITIALIZE/QUEUE/START/PROCESSOR=SEVMS_SMB LPA0:
$ INITIALIZE/QUEUE/START/PROCESSOR=SEVMS$LATSYM LN03:
```

Note: **Any queues initialized without specifying an SEVMS print symbiont name will use the standard VMS print symbiont.**

If the INITIALIZE/QUEUE command is not successful, it may be due to a problem with the template library (SEVMS$SMB_LIB.TLB) or database (SEVMS$SMB_HDRFRM.DAT). If this is the case, in addition to the singl error status line displayed at the terminal by the INITIALIZE/QUEUE command, a more complete error message will be sent to any PRINT operators and to the operator's log file. See the *Guide to Maintaining a VMS System* manual, in the "VMS System Management Subkit" for information about the operator's log file.

The services of the SEVMS print symbiont are accessed the same way as with the standard VMS print symbiont. Users queue documents for printing by using the DCL PRINT command as described in the *VMS DCL Dictionary* and are able to specify all the same qualifier options that the standard symbiont accepts.

In regard to printing, SEVMS operates unlike standard VMS, where protection checking is performed at the time the PRINT command is issued, and is limited to determining that the user can read the file. In addition, the SEVMS print symbiont does protection checks at the time the file is printed and checks to see that the file classification falls within the classification range of the printer. As a result, the user might see a NO PRIVILEGE error returned as a job completion status if the print-time check fails.

7.6 Printing Listing Files

In order to achieve nicely formatted reports with the SEVMS print symbiont, the page size of listings produced by native mode programs will probably need to be adjusted.

VMS print symbionts normally add three lines for the top margin, three lines for the bottom margin, and three lines for heading information. The heading information usually consists of a language-processor identification line, source-program identification line, and one blank line. Given a physical page size of 66 lines (define/form/length=66), there are 57 lines left for text.

In the case of print symbiont templates, the addition of page header and page trailer text lines causes the vertical page size to exceed the physical page size. When this happens, the excess lines overflow onto the next physical form and the symbiont skips to the head of the following form. This causes a listing that has a page of text followed by a few lines on the next page, followed by a page of text, followed by a few lines on the next page, and so on.

Native mode programs can be requested to generate listings with a page size of fewer than 57 lines. This is done by defining the logical name SYS$LP_LINES. This logical name can be defined to be any value in the range of 30 to 255, inclusive, and indicates to the native mode compilers the physical page size to use when computing how many lines of text to place on a page.

For example, if SYS$LP_LINES is defined as 66 (the default), the native mode compilers will generate 57 text lines per page as described above. But, if SYS$LP_LINES is defined as 56, the compilers will generate 47 text lines per page, allowing 10 lines of print symbiont header and trailer text for the listing.

The following example changes the vertical page size for all users to 56 lines per page:

```
$ DEFINE/SYSTEM SYS$LP_LINES 56
```

Individual users may change the vertical page size on a group or process basis.

The number for SYS$LP_LINES should, in general, be 66, minus the number of header lines, minus the number of trailer lines.

7.7 Using Print Symbiont Reset Routines

Using the /SEPARATE=(RESET=(xyz)) qualifier when initializing a print queue can cause a problem with the sevms print symbiont. (For example, the "/SEPARATE=(RESET=(ANSI$RESET))" may be specified on LN03 print queues.)

If the /SEPARATE=(RESET=(...)) qualifier is used, the specified Device Control Library reset routine will be executed *before* the Job Back Pages are output, rather than *after*, as might be expected. Therefore, the Job Back Pages may be printed in a format which is different than that of the preceding file.

For example, if a file was being printed on an LN03 printer in LANDSCAPE mode, the "ANSI$RESET" portion of the qualifier in the example above would cause the Job Back Pages to be printed in PORTRAIT mode. In this case, the problem can be corrected by using a narrow format for the back pages in the template.

If such a solution is not acceptable, the /SEPARATE=(RESET=(...)) qualifier should **not** be used.

A SEVMS_DEFAULT Symbiont Template

This section contains the SEVMS default symbiont template used for processing a classified print job. In the actual default template, the lines span 132 columns and the text is centered.

An example of the output of this template is provided on the following pages.

```
.FRONT
********************************************************************************
**********************  SEVMS SECURE PRINT SYMBIONT  ***************************
********************************************************************************
***************************** SEVMS_DEFAULT ************************************
********************************************************************************

.TRIPLE
START_JOB

.TRIPLE
.FILENAME

.TRIPLE
.SEC_LEV

.DOUBLE
NUMBER: .RANDOM

********************************************************************************
**********************  SEVMS SECURE PRINT SYMBIONT  ***************************
********************************************************************************
.BACK
********************************************************************************
**********************  SEVMS SECURE PRINT SYMBIONT  ***************************
********************************************************************************

.TRIPLE
.FILENAME

.TRIPLE
.SEC_LEV

.TRIPLE
END_JOB

.DOUBLE
NUMBER: .RANDOM

********************************************************************************
***************************** SEVMS_DEFAULT ************************************
********************************************************************************
**********************  SEVMS SECURE PRINT SYMBIONT  ***************************
********************************************************************************
.HEADER
* File:  .FILENAME          Date: .DATE           PAGE: .PAGE of .TPAGE      *
* Classification:  .SEC_LEV  Categories:  .SEC_CAT                           *
********************************************************************************
.TRAILER
********************************************************************************
* Classification:  .SEC_LEV  Categories:  .SEC_CAT                           *
* File:  .FILENAME          Date: .DATE           PAGE: .PAGE of .TPAGE      *
```

Index

A

Access accountability • 3–7
Access check
  of ODS-2 files • 3–5
Access checking
  discretionary • 3–3
  mandatory • 3–2
  of subjects and objects • 3–2
Access Control List
  see ACL
Access mode
  types of • 3–2
Access mode check
  description of • 3–2
Access protection
  of printers • 3–4
Access rights block
  see ARB
Access Rights Block
  see ARB
Account
  establishing classification of • 5–14
ACL
  effect of BYPASS privilege • 5–7
  use in SEVMS • 1–1
  VMS access checking of • 3–3
ADD/IDENTIFIER command
  adding identifiers to rights database • 5–3
Alarm
  auditing features • 6–1
  classification change failure • 6–8
  classification change success • 6–8
  file access • 3–7, 6–6
  file access failure • 6–7
  file access success • 6–8
  label bypass • 6–7
  printed file failure • 6–7
  printed file success • 6–6
  print symbiont • 6–6
Alarm Messages
  SEVMS • 6–6
  VMS • 6–6
ANALYZE/AUDIT command • 6–5 to 6–6
  qualifiers • 6–5
ANSI tape volume
  protection of • 5–9
ARB • 3–1
  of a process • 4–1
Audit archive file
  audit record description • 6–6
Auditing
  alarm • 3–7
  commands and qualifiers • 6–2 to 6–6
  example of configuring print queues • 7–7
  features • 6–1
  introduction to • 6–1
  PASSALL qualifier use • 7–6
Audit record
  examples of • 6–6
Audit record format • 6–9
AUTHORIZE utility
  ADD/IDENTIFIER command • 5–4
  adding classified users • 5–14
  adding identifiers to rights database • 5–3
  example of adding identifiers with • 5–5
  security manager's use of • 5–1
Aysynchronous serial line interface
  in relation to classifying terminals • 5–16

B

.BACKPAGE directive
  considerations concerning use • 7–5
  description of • 7–9
Back page element • 7–8
BACKUP utility
  backup of tape devices • 5–9
  classifying volumes • 5–10
  description of SEVMS version of • 5–24
  using for file classification • 5–10
Batch job
  classification of • 4–2
  propagation of classification • 3–4
Binary audit record format • 6–9
BYPASS privilege
  creating directories • 4–5
  description of • 5–2
  effect on classification • 5–7
  effect upon access to objects • 3–4

BYPASS privilege (cont'd.)
use with MAIL utility • 5–24

C

CAPTIVE account
logging in to • 4–2
Category
see Secrecy category
see Integrity category • 2–6
CATEGORY qualifier
adding identifiers to rights database • 5–5
Classification
changing • 5–6
definition of • 2–1
description of • 4–3 to 4–6
displaying • 4–2
example of adding multi-class user • 5–15
example of adding single class user • 5–15
in relation to networks • 5–19
modifying files • 3–6
of accounts • 5–14
of batch jobs • 4–2
of captive account • 4–2
of detached process • 4–2
of devices • 5–8
of directories • 5–11
of disk devices • 5–9
of disk volumes • 5–7
of files • 5–7
of FILES-11 volumes • 5–9
of interactive log-ins • 5–13
of logical name tables • 5–12
of mailboxes • 5–13
of network process • 4–2
of object • 4–3
of objects • 2–6, 3–4, 5–7
of ports • 5–17
of process • 5–14
of process at log-in
default • 4–2
description of • 4–1
error messages • 4–1
example of • 4–1
of queues • 5–12
of session • 4–3
of SYSTEM account • 5–16
of tape devices • 5–9
of tape drives • 5–7
of terminals • 5–7
Classification (cont'd.)
of terminals connected by DECNET • 5–16
of terminals connected by serial line interface • 5–16
of terminal servers • 5–17
of user • 2–2
of VWS windows • 5–8
privileges during log-in • 5–18
ranges
see Classification range
rules of propagation • 3–4
Classification block
see also Classification label
see also Classification string
see also Label
definition (of) in relation to labels • 2–3
Classification category
description of • 2–6
Classification change failure alarm
description of • 6–8
Classification change success alarm
description of • 6–8
Classification label
description of • 2–3
effect of BACKUP • 5–24
Classification level
description of • 2–5
numeric value • 2–5
text identifier • 2–5
types of • 2–5
Classification range
definition of • 2–3
description of • 2–6
example of implementing • 5–15
in relation to printer access protection • 3–4
of user account • 4–1
of volumes • 5–9
syntax example • 2–6
Classification string
description of • 2–4
in classification labels • 2–3
input strings
description of • 2–4
examples of • 2–4
output string
description of • 2–7
examples of • 2–7
format of • 2–7
see also Label • 2–3
types of • 2–4
CLASS_PROT parameter

CLASS_PROT parameter (cont'd.)
classifying files • 5–10
enabling and disabling mandatory controls • 5–6
Clearance
definition of • 2–1
of user • 2–2
Command syntax
of SEVMS commands • ix
of VMS commands • ix
Complementary security techniques
see Security features
Confidentiality of data • 2–2
Conventions
SEVMS • ix
VMS • viii
CREATE/DIRECTORY command • 4–5
$CREMBX system service
classifying mailboxes • 5–13
Critical object
protection of • 5–7

D

Data
confidentiality of • 2–2
sensitivity of • 5–2
DECnet
in classified environment • 5–19
in relation to classifying terminals • 5–16
DECwindows session
logging into • 4–2
Default symbiont template
example of • A–1
DEFINE/SYSTEM command • 7–20
Descriptive name
description of • 4–6
Detached process
classification of • 4–2
Device
as an object • 2–2
classification of • 5–8
DEVICE object type • 4–3, 5–7
DEVOUR privilege
description of • 5–2
DIGITAL consultant • v
Directive
element • 7–8
field • 7–9
format • 7–12
Directive (cont'd.)
notes about usage • 7–13
Directory
creation (of) by privileged user • 4–5
creation (of) by unprivileged user • 4–5
protection of • 5–11
DIRECTORY/FULL command • 4–4
DIRECTORY/SECURITY command • 4–4
Directory structure
organization of • 4–4
Discretionary access
effect of BYPASS privilege • 5–7
introduction to • 1–1
Discretionary access check
description of • 3–3
Disk device
use of MOUNT command • 5–9
Disk drive
classification of • 5–8
SEVMS support of • 3–1
Disk volume
BACKUP of • 5–24
classification of • 5–7
SEVMS support of • 3–1
use of MOUNT command • 5–9
Disk volume classification
security manager's role • 5–1
Documentation
associated SEVMS manuals • vi
associated VMS manuals • vii, viii
relationship of VMS and SEVMS manuals • viii
Dominate
definition of • ix
in mandatory access check • 3–2
Downgrade
of information • 5–21
DOWNGRADE keyword
of SET AUDIT command • 6–2
DOWNGRADE privilege
auditing use of • 6–1
creating directories • 4–5
description of • 5–2
effect on classification • 5–6
effect upon access to objects • 3–4
use with MAIL utility • 5–24
Dynamic classification
description of • 5–7